

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO



BÁO CÁO
ĐỒ ÁN CUỐI KỲ

Đề tài: CÂY NHỊ PHÂN TÌM KIẾM (BINARY SEARCH TREE) VÀ ỨNG DỤNG ĐỂ TẠO TỪ ĐIỂN ANH – VIỆT

LỚP HỌC PHẦN: PROJ215879_22_1_13CLC
HỌC KỲ I – NĂM HỌC 2022 – 2023

Thực hiện:

Huỳnh Thanh Tuấn – 20110120

Trần Nguyễn Duy Linh – 20110516

Giảng viên hướng dẫn: Th.S Nguyễn Quang Ngọc

Thành phố Hồ Chí Minh, Tháng 11 năm 2022

MỤC LỤC

DANH MỤC HÌNH ẢNH.....	2
DANH MỤC BẢNG.....	2
I. Đặt tả	3
II. Phân công công việc	3
III. Thiết kế	5
1. Thuật toán	5
2. Thiết kế lớp.....	10
3. Thiết kế FILE.....	24
4. Thiết kế giao diện.....	25
IV. Cài đặt và kiểm thử	26
V. Kết luận	27
VI. Tài liệu tham khảo	28

DANH MỤC HÌNH ẢNH

Ảnh 1 - Nạp chồng	5
Ảnh 2 - Chuyển thành chữ thường	6
Ảnh 3 - Tạo đối tượng	6
Ảnh 4 - Huỷ đối tượng	6
Ảnh 5 - Tạo cây con	7
Ảnh 6 - Tìm nút thứ N theo thứ tự trước.....	7
Ảnh 7 - Đếm số lượng nút trong cây.....	7
Ảnh 8 - Tìm kiếm theo ký tự.....	8
Ảnh 9 - Duyệt sâu trước (DFS)	8
Ảnh 10 - Xoá vùng nhớ đệm.....	9
Ảnh 11 - Lấy ký tự chữ/số đầu tiên.....	9
Ảnh 12 - Xoá cò lỗi.....	9
Ảnh 13 - Ghi vào file.....	10

DANH MỤC BẢNG

Bảng 1: Bảng phân công	3
Bảng 2: Kế hoạch thực hiện	4
Bảng 3: Danh mục các lớp	10
Bảng 4: Phương thức lớp TuNgu	11
Bảng 5: Phương thức lớp Node.....	11
Bảng 6: Phương thức lớp Node_list.....	12
Bảng 7: Phương thức lớp Queue	12
Bảng 8: Phương thức lớp Stack.....	13
Bảng 9: Phương thức public lớp BST	13
Bảng 10: Phương thức private lớp BST	18
Bảng 11: Phương thức lớp APP	22
Bảng 12: Danh sách các giao diện chương trình.....	25
Bảng 13: Kiểm thử chương trình	26

I. Đặt tả

Từ việc ôn tập lại các kiến thức về cây nhị phân tìm kiếm (Binary Search Tree) như cách hoạt động, các thuật toán duyệt cây, thêm xóa các node trên cây, ... từ đó ứng dụng vào để tạo một từ điển Anh – Việt.

- Từ điển này được sử dụng để tra cứu các thông tin của từ ngữ tiếng Anh như từ loại, nghĩa, ví dụ sử dụng.
- Dữ liệu đầu vào:
 - do người dùng nhập từ bàn phím
- Các tính năng:
 - tra cứu từ ngữ
 - thêm, chỉnh sửa, xóa từ
- Giao diện sử dụng: ứng dụng được viết bằng ngôn ngữ C++, người dùng tương tác với chương trình thông qua command line.

II. Phân công công việc

Bảng 1: Bảng phân công

Tên sinh viên	Mô tả khái quát mảng công việc SV thực hiện trong đề án	Ước tính phần trăm đóng góp
Huỳnh Thanh Tuấn	<ul style="list-style-type: none">• Thiết kế giao diện• Thiết kế các hàm xử lý trên cây nhị phân: duyệt, thêm, xóa nút khi biết key.• Viết báo cáo	100%
Trần Nguyễn Duy Linh	<ul style="list-style-type: none">• Thiết kế các hàm xử lý trên cây nhị phân: xóa nút thứ n, tìm kiếm.• Viết báo cáo	100%

Bảng 2: Kế hoạch thực hiện

STT	Mục tiêu	Quá trình thực hiện										
1	Hiểu yêu cầu.	o	o									
2	Mô tả các yêu cầu của đề án.	o	o									
3	Tìm hiểu GitHub	o	o	o								
4	Ôn tập ngôn ngữ C++	o	o	o								
5	Ôn tập và tìm hiểu các giải thuật liên quan đến cây, con trỏ	o	o	o								
6	Tìm các thuật toán duyệt cây		o	o	o	o						
7	Viết các thuật toán duyệt cây, thao tác trên cây					o	o	o	o	o	o	
8	Xây dựng ứng dụng.						o	o	o	o	o	o
9	Thiết kế giao diện console cơ bản.								o	o	o	o
10	Thiết kế class.		o	o	o	o	o	o	o	o	o	o
11	Thực hiện chương trình.						o	o	o	o	o	o
12	Tối ưu hoá và sửa mã code.									o	o	o
13	Thử nghiệm.									o	o	o
14	Họp thảo luận trực tuyến	o	o	o	o	o	o	o	o	o	o	o
15	Viết báo cáo.	o	o	o	o	o	o	o	o	o	o	o
Tuần		3	4	5	6	7	8	9	10	11	12	13

III. Thiết kế

1. Thuật toán

Các thuật toán:

+ Thực hiện nạp chồng lại các toán tử <, !=, << để có thể thực hiện so sánh 2 đối tượng thuộc lớp TuNgu và xuất đối tượng ra bình thường. Tại phương thức nạp chồng << sử dụng hàm bạn để có thể truyền tham số đầu tiên là khác với tham số mặc định là đối tượng TuNgu và trả về kiểu tham chiếu giúp tiếp tục thực hiện việc xuất ra màn hình tại vị trí toán tử << được gọi.

```
bool operator<(TuNgu B)
{
    char a[21], b[21];
    ChuyenChuThuong(this->TiengAnh, a);
    ChuyenChuThuong(B.TiengAnh, b);
    if (strcmp(a, b) < 0)
    {
        return true;
    }
    return false;
}

bool operator!=(TuNgu B)
{
    char a[21], b[21];
    ChuyenChuThuong(this->TiengAnh, a);
    ChuyenChuThuong(B.TiengAnh, b);
    if (strcmp(a, b) != 0)
    {
        return true;
    }
    return false;
}

friend ostream &operator<<(ostream &os, TuNgu data)
{

```

Ảnh 1 - Nạp chồng

+ Cần phải chuyển thành chữ thường trước khi so sánh do người dùng có thể nhập cả ký tự in hoa lẫn thường nên khi thực hiện so sánh cần phải chuyển tất cả các ký tự về chữ thường giúp việc so sánh được nhất quán.

```

void ChuyenChuThuong(char A[], char kq[])
{
    int i;
    for (i = 0; i < strlen(A); i++)
    {
        if (A[i] >= 'A' && A[i] <= 'Z')
        {
            kq[i] = A[i] + 32;
        }
        else
        {
            kq[i] = A[i];
        }
    }
    kq[i] = '\0';
}

```

Ảnh 2 - Chuyển thành chữ thường

+ Tại các lớp phải thực hiện xây dựng phương thức hàm tạo riêng không sử dụng hàm tạo mặc định là do khi sử dụng malloc để tạo ra đối tượng lúc này hàm tạo không được gọi tự động. Vì vậy cần phải gọi hàm tạo từ đối tượng sau khi thực hiện cấp phát động.

```

Node_list<T> *p;
p = (Node_list<T> *)malloc(sizeof(Node_list<T>));
if (p == NULL)
{
    cout << "\nCo loi! khong the cap phat bo nho." << endl;
    return 0;
}
p->Creat_Node_list(value);

```

Ảnh 3 - Tạo đối tượng

+ Hàm hủy đối tượng giúp thu hồi các vùng nhớ được cấp phát động phải thực hiện gọi đệ quy theo thứ tự sau giúp đảm bảo thu hồi vùng nhớ của các nút con trước khi thu hồi vùng nhớ của nút cha.

```

void DESTROY_TREE(Node<T> *&root)
{
    if (root != NULL)
    {
        DESTROY_TREE(root->left);
        DESTROY_TREE(root->right);
        free(root);
    }
}

```

Ảnh 4 - Hủy đối tượng

+ Thực hiện tạo cây mới từ node hiện tại thực hiện đệ quy theo thứ tự trước giúp đảm bảo nút gốc là node hiện tại

```

void Copy_Node(Node<T> *&Roottemp, Node<T> *temp)
{
    if (temp != NULL)
    {
        TREE_INSERT(Roottemp, temp->data);
        Copy_Node(Roottemp, temp->left);
        Copy_Node(Roottemp, temp->right);
    }
}

```

Ảnh 5 - Tạo cây con

+ Thực hiện truyền theo kiểu tham chiếu biến flag trong phương thức NthPreordernode vì khi thực hiện đệ qui giá trị flag được cập nhật nên khi trả về cần lưu lại giá trị này để biết được đã đến nút thứ N không. Khi flag bằng N biến temp được gán giá trị nên khi thoát khỏi phương thức giá trị temp cần được cập nhật nên biến temp được truyền theo kiểu tham chiếu.

```

void NthPreordernode(Node<T> *R, int N, Node<T> *&temp, int &flag)
{
    if (R != NULL)
    {
        if (flag == N)
        {
            temp = R;
        }
        flag = flag + 1;
        NthPreordernode(R->left, N, temp, flag);
        NthPreordernode(R->right, N, temp, flag);
    }
}

```

Ảnh 6 - Tìm nút thứ N theo thứ tự trước

+ Tại phương thức SOLUONG_NODE biến temp được truyền theo kiểu tham chiếu giúp khi thực hiện xong lệnh đệ qui giá trị tại vùng nhớ của biến temp được cập nhật.

```

void SOLUONG_NODE(Node<T> *root, int &kq)
{
    if (root != NULL)
    {
        SOLUONG_NODE(root->left, kq);
        kq += 1;
        SOLUONG_NODE(root->right, kq);
    }
}

```

Ảnh 7 - Đếm số lượng nút trong cây

+ Tại phương thức TREE_SEARCH_CHAR vì chuỗi từ tiếng anh ký tự đầu tiên có thể là chữ cái in Hoa hoặc thường. Do ký tự tìm kiếm đã được chuyển thành chữ thường trước khi truyền vào nếu ký tự đầu tiên của từ hiện tại là ký tự in hoa

thực hiện cộng thêm 32 vào để có được giá trị chữ thường của chữ in hoa hiện tại

```
void TREE_SEARCH_CHAR(Node<TuNgu> *root, char k)
{
    if (root != NULL)
    {
        TREE_SEARCH_CHAR(root->left, k);
        if (root->data.TiengAnh[0] >= 'A' && root->data.TiengAnh[0] <= 'Z')
        {
            if ((root->data.TiengAnh[0]) + 32 == k)
            {
                for (int i = 0; i < strlen(root->data.TiengAnh); i++)
                {
                    if (root->data.TiengAnh[i] >= 'A' && root->data.TiengAnh[i] <= 'Z')
                    {

```

Ảnh 8 - Tìm kiếm theo ký tự

+ Tại thuật toán duyệt theo chiều sâu trước thực hiện thêm node phải trước node trái. Để khi thực hiện node trái được lấy ra đầu tiên hay cây con bên trái được duyệt trước cây con bên phải.

```
// Thuật toán duyệt sau trước DFS
void DFS(Node<T> *Root)
{
    Stack<T> NganXep;
    Node<T> temp;
    if (Root == NULL)
        return;
    NganXep.PushS(*Root);
    while (NganXep.sp != NULL)
    {
        NganXep.PopS(&temp);
        cout << temp.data << "\t";
        if (temp.right != NULL)
        {
            NganXep.PushS(*(temp.right));
        }
        if (temp.left != NULL)
        {
            NganXep.PushS(*(temp.left));
        }
    }
}
```

Ảnh 9 - Duyệt sâu trước (DFS)

+ Tại thuật toán xóa bộ nhớ tạm khi người dùng nhập vào giúp xóa hết tất cả các ký tự còn lại sau khi dữ liệu đã được đọc hoàn tất. Giúp cho việc lấy ký tự người dùng nhập vào tiếp theo không bị bỏ qua do còn ký tự trong vùng nhớ tạm này nên hàm nhập sẽ thực hiện lấy ra mà không cần người dùng nhập vào lần tiếp theo. Bên cạnh đó phương thức này giúp phát hiện người dùng nhập vào có khoảng cách giữa các từ.

```

int Xoa_Vung_Nho_Tam()
{
    char temp;
    int trangthai = 0;
    temp = cin.get();
    if (temp == '\n')
    {
        trangthai = 0;
    }
    while (temp != '\n')
    {
        if (temp != ' ' && temp != '\t')
            trangthai = 1;
        temp = cin.get();
    }
    return trangthai;
}

```

Ảnh 10 - Xoá vùng nhớ đệm

+ Thực hiện bỏ qua các ký tự khoảng trắng, tab, enter trước ký tự đầu tiên khi nhập vào qua thuật toán

```

while (temp == '\n' || temp == ' ' || temp == '\t')
{
    temp = cin.get();
}

```

Ảnh 11 - Lấy ký tự chữ/số đầu tiên

+ Tại phương thức Nhap thực hiện lệnh cin.clear giúp xoá cò lỗi khi dữ liệu nhập vào không đúng với kiểu của biến lưu giá trị. Nhờ thực hiện lệnh xoá cò lỗi nên lệnh lấy dữ liệu vào tiếp theo có thể thực hiện được.

```

while (!(cin >> x))
{
    cout << "Co loi. Vui long chi nhap so nguyen" << endl;
    cout << "Nhap lai: ";
    cin.clear();
    Xoa_Vung_Nho_Tam();
};
cout << "Ban vua nhap vao so: " << x << endl;
break;

```

Ảnh 12 - Xoá cò lỗi

+ Tại phương thức Ghi_txt thực hiện ghi theo thứ tự trước giúp cho nút gốc của cây được ghi đầu tiên. Giúp cho khi thực hiện đọc file nút đầu tiên được đọc là nút gốc. Từ đó giúp đảm bảo được cấu trúc của cây như lúc ghi file.

```

void Ghi_txt(Node<TuNgu> *T, FILE *f)
{
    static char temp = '\n';
    if (T != NULL)
    {
        fwrite(T->data.TiengAnh, sizeof(T->data.TiengAnh), 1, f);
        fwrite(&temp, sizeof(temp), 1, f);
        fwrite(T->data.TiengViet, sizeof(T->data.TiengViet), 1, f);
        fwrite(&temp, sizeof(temp), 1, f);
        fwrite(T->data.Tuloai, sizeof(T->data.Tuloai), 1, f);
        fwrite(&temp, sizeof(temp), 1, f);
        fwrite(T->data.Vidu, sizeof(T->data.Vidu), 1, f);
        fwrite(&temp, sizeof(temp), 1, f);
        Ghi_txt(T->left, f);
        Ghi_txt(T->right, f);
    }
}

```

Ảnh 13 - Ghi vào file

2. Thiết kế lớp

Danh mục lớp:

Bảng 3: Danh mục các lớp

TT	Tên lớp	Mục đích	Người thực hiện
1	TuNgu	Để tạo kiểu dữ liệu tên TuNgu cho ứng dụng từ điển	Huỳnh Thanh Tuấn
2	Node	Để đặt tả cấu trúc của một nút trong cây nhị phân tìm kiếm	Huỳnh Thanh Tuấn
3	Node_list	Để đặt tả cấu trúc của một node trong danh sách liên kết đơn	Trần Nguyễn Duy Linh
4	Queue	Dùng để đặt tả cấu trúc dữ liệu hàng đợi sử dụng danh sách liên kết đơn	Trần Nguyễn Duy Linh
5	Stack	Dùng để đặt tả cấu trúc dữ liệu ngăn xếp sử dụng danh sách liên kết đơn	Huỳnh Thanh Tuấn
6	BST	Dùng để đặt tả cây nhị phân tìm kiếm	Huỳnh Thanh Tuấn Trần Nguyễn Duy Linh
7	APP	Dùng để đặt tả thiết kế giao diện cho ứng dụng.	Huỳnh Thanh Tuấn

Danh mục phương thức:

Bảng 4: Phương thức lớp TuNgu

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Create_TuNgu() Input: Không có Output: Không có Pseudo code: Không có	Khởi tạo dữ liệu ban đầu cho đối tượng.	(33)	Huỳnh Thanh Tuấn
2	operator<(TuNgu B) Input: B Output: true/false Pseudo code: Không có	Dùng để so sánh nhỏ hơn giữa hai đối tượng được tạo ra từ lớp TuNgu	(43)	Huỳnh Thanh Tuấn
3	operator!=(TuNgu B) Input: B Output: true/false Pseudo code: không có	Dùng để so sánh khác giữa hai đối tượng được tạo ra từ lớp TuNgu	(54)	Huỳnh Thanh Tuấn
4	operator<<(ostream &os, TuNgu data) Input: os, data Output: os Pseudo code: không có	Dùng để xuất ra màn hình đối tượng được tạo ra từ lớp TuNgu	(65)	Huỳnh Thanh Tuấn
5	ChuyenChuThuong(char A[], char kq[]) Input: A[], kq[] Output: không có Pseudo code: không có	Dùng để sao chép A sang kq dưới dạng chữ thường.	(106)	Huỳnh Thanh Tuấn

Bảng 5: Phương thức lớp Node

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Create_Node() Input: không có	Dùng để khởi tạo giá trị ban đầu	(137)	Huỳnh Thanh Tuấn

	Output: không có Pseudo code: không có	cho đối tượng của lớp Node		
--	---	-------------------------------	--	--

Bảng 6: Phương thức lớp Node_list

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Creat_Node_list(Node<T> data) Input: data Output: không có Pseudo code: không có	Dùng để khởi tạo giá trị ban đầu cho đối tượng của lớp Node_list	(158)	Trần Nguyễn Duy Linh

Bảng 7: Phương thức lớp Queue

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Create_Queue() Input: không có Output: không có Pseudo code: không có	Dùng để khởi tạo giá trị ban đầu cho đối tượng của lớp Queue	(178)	Trần Nguyễn Duy Linh
2	PushQ(Node<T> value) Input: value Output: int Pseudo code: Nếu hàng đợi rỗng thì con trỏ ở đỉnh và cuối hàng đợi cùng trỏ đến giá trị mới. Ngược lại con trỏ cuối hàng đợi trỏ đến giá trị mới	Dùng để thêm một node vào cuối hàng đợi	(184)	Trần Nguyễn Duy Linh
3	PopQ(Node<T> *value) Input: value Output: int Pseudo code: Nếu hàng đợi khác rỗng lấy giá trị ở đỉnh hàng đợi	Dùng để lấy một node ở đỉnh hàng đợi	(207)	Trần Nguyễn Duy Linh

Bảng 8: Phương thức lớp Stack

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Create_Stack() Input: không có Output: không có Pseudo code: không có	Dùng để khởi tạo giá trị ban đầu cho đối tượng Stack	(238)	Huỳnh Thanh Tuấn
2	PushS(Node<T> value) Input: value Output: int Pseudo code: không có	Dùng để thêm 1 node vào đỉnh ngăn xếp	(243)	Huỳnh Thanh Tuấn
3	PopS(Node<T> *value) Input: value Output: int Pseudo code: Nếu ngăn xếp khác rỗng lấy giá trị ở đỉnh.	Dùng để lấy ra 1 node ở đỉnh ngăn xếp	(258)	Huỳnh Thanh Tuấn

Bảng 9: Phương thức public lớp BST

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Create_BST() Input: Không có Output: Không có Pseudo code: Không có	Dùng để khởi tạo giá trị ban đầu cho đối tượng từ lớp BST	(288)	Huỳnh Thanh Tuấn
2	DestroyTree() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương DESTROY_TREE	(293)	Huỳnh Thanh Tuấn
3	GetRoot() Input: Không có Output: Node<T> * Pseudo code: Không có	Dùng để lấy thuộc tính Root của cây BST	(299)	Huỳnh Thanh Tuấn

4	InsertTree(T value) Input: value Output: int Pseudo code: Không có	Dùng để gọi phương thức TREE_INSERT	(304)	Huỳnh Thanh Tuấn
5	Timkey(T key) Input: key Output: Node<T> * Pseudo code: Không có	Dùng để gọi phương thức TREE_SEARCH	(309)	Huỳnh Thanh Tuấn
6	XoaKey(T key) Input: key Output: Node<T> * Pseudo code: Tìm key trong cây nếu có thực hiện xoá	Phối hợp phương thức TREE_SEARCH và TREE_DELETE Để xoá node có key truyền vào	(314)	Huỳnh Thanh Tuấn
7	DuyetTruoc() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức PREORDER_TREE_WALK	(323)	Huỳnh Thanh Tuấn
8	DuyetGiua() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức INORDER_TREE_WALK	(328)	Huỳnh Thanh Tuấn
9	DuyetSau() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức POSTORDER_TREE_WALK	(333)	Huỳnh Thanh Tuấn
10	DuyetRongTruoc() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức BFS	(338)	Huỳnh Thanh Tuấn

11	DuyetSauTruoc() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức DFS	(343)	Huỳnh Thanh Tuấn
12	Clone_Tree(Node<T> *temp) Input: temp Output: Không có Pseudo code: Không có	Dùng để gán temp vào thuộc tính Root	(348)	Huỳnh Thanh Tuấn
13	ThongTinCay() Input: Không có Output: Không có Pseudo code: Không có	Dùng để gọi phương thức INFO_TREE	(353)	Huỳnh Thanh Tuấn
14	Slg_Node(int &kq) Input: kq Output: Không có Pseudo code: Không có	Dùng để gọi phương thức SOLUONG_NODE	(358)	Huỳnh Thanh Tuấn
15	Value_Node_Truoc(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt trước nếu có in node đó ra màn hình	Dùng để in giá trị node thứ N theo thứ tự trước	(363)	Trần Nguyễn Duy Linh
16	Value_Node_Giua(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt giữa nếu có in node đó ra màn hình	Dùng để in giá trị node thứ N theo thứ tự giữa	(376)	Trần Nguyễn Duy Linh

17	Value_Node_Sau(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt sau nếu có in node đó ra màn hình	Dùng để in giá trị node thứ N theo thứ tự sau	(389)	Trần Nguyễn Duy Linh
18	Del_Node_Truoc(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt trước nếu có xoá node đó	Dùng để xoá giá trị node thứ N theo thứ tự trước	(403)	Trần Nguyễn Duy Linh
19	Del_Node_Giua(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt giữa nếu có xoá node đó	Dùng để xoá giá trị node thứ N theo thứ tự giữa	(402)	Trần Nguyễn Duy Linh
20	Del_Node_Sau(int N) Input: N Output: Không có Pseudo code: tìm node thứ N theo thứ tự duyệt sau nếu có xoá node đó	Dùng để xoá giá trị node thứ N theo thứ tự sau	(437)	Trần Nguyễn Duy Linh
21	SaoChepPreorder_N(int N) Input: N Output: Node<T> * Pseudo code: tìm node thứ N theo thứ tự trước nếu có	Dùng để tạo cây con có gốc là node thứ n theo thứ tự trước	(455)	Trần Nguyễn Duy Linh

	tạo cây mới với node tìm được là gốc			
22	SaoChepInorder_N(int N) Input: N Output: Node<T> * Pseudo code: tìm node thứ N theo thứ tự giữa nếu có tạo cây mới với node tìm được là gốc	Dùng để tạo cây con có gốc là node thứ n theo thứ tự giữa	(470)	Trần Nguyễn Duy Linh
23	SaoChepPostorder_N(int N) Input: N Output: Node<T> * Pseudo code: tìm node thứ N theo thứ tự sau nếu có tạo cây mới với node tìm được là gốc	Dùng để tạo cây con có gốc là node thứ n theo thứ tự sau	(485)	Trần Nguyễn Duy Linh
24	IN_THEO_CHAR(char key) Input: key Output: Không có Pseudo code: Không có	Dùng để gọi phương thức TREE_SEARCH_CHARACTER để tìm theo ký tự	(503)	Huỳnh Thanh Tuấn
25	ThôngTinNode_TN(T key) Input: key Output: int Pseudo code: Không có	Dùng để gọi phương thức TREE_SEARCH In ra thông tin của node có giá trị bằng key	(512)	Huỳnh Thanh Tuấn

Bảng 10: Phương thức private lớp BST

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	TREE_INSERT(Node<T> *&Root, T value) Input: Root, value Output: int Pseudo code: Tạo node mới với giá trị value. Nếu value có giá trị nhỏ hơn giá trị của node Root thì thêm node mới vào cây con bên trái ngược lại thêm vào cây con bên phải	Dùng để thêm node cho cây nhị phân tìm kiếm	(530)	Huỳnh Thanh Tuấn
2	TREE_SEARCH(Node<T> *&root, T key) Input: root, key Output: Node<T> * Pseudo code: So sánh giá key với node hiện tại nếu nhỏ hơn thì tìm bên cây con trái ngược lại tìm bên cây con phải của node hiện tại	Dùng để tìm kiếm key trong cây	(560)	Huỳnh Thanh Tuấn
3	TREE_MINIMUM(Node <T> *temp) Input: temp Output: Node<T> * Pseudo code: Không có	Dùng để tìm kiếm giá trị nhất trong cây truyền vào	(572)	Huỳnh Thanh Tuấn
4	TREE_SUCCESSOR(Node <T> *temp) Input: temp Output: Node<T> *	Dùng để tìm node liền kề có giá trị lớn hơn node truyền vào	(580)	Huỳnh Thanh Tuấn

	Pseudo code: Nếu node truyền vào có cây con bên phải thì tìm giá trị nhỏ nhất của cây con bên phải ngược lại tìm tổ tiên đầu tiên của temp có key lớn hơn temp			
5	TREE_DELETE(Node<T> * &root, Node<T> *z) Input: root, z Output: Node<T> * Pseudo code: Nếu Node cần xoá không có cây con hoặc có 1 cây con thực hiện xoá node đó ngược lại thực hiện tìm successor gán vào vị trí node cần xoá	Dùng để xoá node truyền vào trong cây	(593)	Huỳnh Thanh Tuấn
6	PREORDER_TREE_WALK(Node<T> *R) Input: R Output: không có Pseudo code: không có	Dùng để duyệt cây theo thứ tự trước	(626)	Huỳnh Thanh Tuấn
7	INORDER_TREE_WALK(Node<T> *R) Input: R Output: không có Pseudo code: không có	Dùng để duyệt cây theo thứ tự giữa	(636)	Huỳnh Thanh Tuấn
8	POSTORDER_TREE_WALK(Node<T> *R) Input: R Output: không có Pseudo code: không có	Dùng để duyệt cây theo thứ tự sau	(646)	Huỳnh Thanh Tuấn

9	BFS(Node<T> *Root) Input: Root Output: không có Pseudo code: Lấy node ở đỉnh hàng đợi nếu Node có Node bên trái hoặc phải thực hiện thêm Node đó vào hàng đợi	Dùng để duyệt cây theo chiều rộng trước	(656)	Huỳnh Thanh Tuấn
10	DFS(Node<T> *Root) Input: Root Output: không có Pseudo code: Lấy node ở đỉnh ngăn xếp nếu Node có Node bên trái hoặc phải thực hiện thêm Node đó vào ngăn xếp	Dùng để duyệt cây theo chiều sâu trước	(678)	Huỳnh Thanh Tuấn
11	INFO_TREE(Node<T> *root) Input: root Output: không có Pseudo code: không có	Dùng để in cách vẽ cây	(700)	Huỳnh Thanh Tuấn
12	NthPreordernode(Node<T> > *R, int N, Node<T> *&temp, int &flag) Input: R, N, temp, flag Output: không có Pseudo code: nếu node hiện tại là node thứ N thực hiện lưu node này lại	Dùng để tìm node thứ N theo thứ tự duyet trước	(724)	Trần Nguyễn Duy Linh

13	NthInordernode(Node<T> *R, int N, Node<T> *&temp, int &flag) Input: R, N, temp, flag Output: không có Pseudo code: nếu node hiện tại là node thứ N thực hiện lưu node này lại	Dùng để tìm node thứ N theo thứ tự duyệt giữa	(737)	Trần Nguyễn Duy Linh
14	NthPostordernode(Node<T> *R, int N, Node<T> *&temp, int &flag) Input: R, N, temp, flag Output: không có Pseudo code: nếu node hiện tại là node thứ N thực hiện lưu node này lại	Dùng để tìm node thứ N theo thứ tự duyệt sau	(750)	Trần Nguyễn Duy Linh
15	Copy_Node(Node<T> *&Roottemp, Node<T> *temp) Input: Roottemp, temp Output: không có Pseudo code: không có	Dùng để sao chép cây	(764)	Trần Nguyễn Duy Linh
16	SOLUONG_NODE(Node<T> *root, int &kq) Input: root, kq Output: không có Pseudo code: không có	Dùng để đếm số lượng node trong cây	(774)	Huỳnh Thanh Tuấn
17	TREE_SEARCH_CHAR(Node<TuNgu> *root, char k) Input: root, k	Dùng để in ra màn hình từ tiếng anh theo ký tự đầu tiên	(785)	Huỳnh Thanh Tuấn

	Output: không có Pseudo code: không có			
18	DESTROY_TREE(Node< T> *&root) Input: root Output: không có Pseudo code: không có	Dùng để giải phóng vùng nhớ của các node đã cấp phát trong cây	(824)	Huỳnh Thanh Tuấn

Bảng 11: Phương thức lớp APP

TT	Phương thức	Mục đích	Vị trí	Thực hiện
1	Xoa_Vung_Nho_Tam() Input: không có Output: int Pseudo code: đọc các ký tự từ vùng nhớ nhập nếu tất cả đều là ký tự xuống dòng, tab, cách trả về 0. Ngược lại trả về 1	Dùng để xóa các dữ liệu thừa khi nhập vào	(844)	Huỳnh Thanh Tuấn
2	Nhap_TuVung(char A[], int slg) Input: A[], slg Output: int Pseudo code: Đọc các ký tự người dùng nhập vào nếu xuất hiện các ký tự khác chữ thực hiện nhập lại.	Dùng để kiểm tra và đọc các ký tự chữ người dùng nhập vào	(862)	Huỳnh Thanh Tuấn
3	Nhap_char_so(char &temp) Input: temp Output: không có	Dùng để kiểm tra và đọc 1 ký tự số người dùng nhập vào	(910)	Huỳnh Thanh Tuấn

	Pseudo code: Đọc các ký tự người dùng nhập vào nếu xuất hiện các ký tự khác số thực hiện nhập lại.			
4	KieuT() Input: không có Output: char Pseudo code: không có	Dùng để đọc lựa chọn kiểu dữ liệu của người dùng	(935)	Huỳnh Thanh Tuấn
5	Nhap(char T, int &x, float &y, char &z) Input: T, x, y, z Output: int Pseudo code: Lấy dữ liệu người dùng nhập vào nếu không đúng kiểu cần lưu thực hiện xóa cò lỗi và vùng nhớ tạm	Dùng để kiểm tra và đọc dữ liệu người dùng nhập vào.	(960)	Huỳnh Thanh Tuấn
6	Ghi_txt(Node<TuNgu> *T, FILE *f) Input: T, f Output: không có Pseudo code: Thực hiện ghi Node hiện tại vào file và gọi đệ quy lần lượt cây con bên trái và phải của node hiện tại	Thực hiện ghi dữ liệu vào file	(1021)	Huỳnh Thanh Tuấn
7	Ghifile_txt(Node<TuNgu> *T) Input: T Output: không có Pseudo code: không có	Thực hiện mở file và gọi phương thức Ghi_txt	(1038)	Huỳnh Thanh Tuấn

8	Docfile_txt(BST<TuNgu>&TREE) Input: TREE Output: không có Pseudo code: Thực hiện đọc dòng đầu tiên trong file nếu thành công thực hiện đọc đến khi đủ 4 dòng và thêm dữ liệu vào cây	Dùng để đọc dữ liệu từ file	(1052)	Huỳnh Thanh Tuấn
9	Lua_Chon_Duyet() Input: không có Output: không có Pseudo code: không có	Dùng để bắt phím người dùng nhấn vào	(1086)	Huỳnh Thanh Tuấn
10	GiaoDienChinh() Input: không có Output: không có Pseudo code: không có	Dùng để tạo giao diện chính cho chương trình	(1108)	Huỳnh Thanh Tuấn
11	GiaoDienSoMot() Input: không có Output: không có Pseudo code: không có	Dùng để tạo giao diện thao tác với cây nhị phân kiểu T	(1154)	Huỳnh Thanh Tuấn
12	GiaoDienSoHai() Input: không có Output: không có Pseudo code: không có	Dùng để tạo giao diện thao tác truy cập từ điển	(1928)	Huỳnh Thanh Tuấn

3. Thiết kế FILE

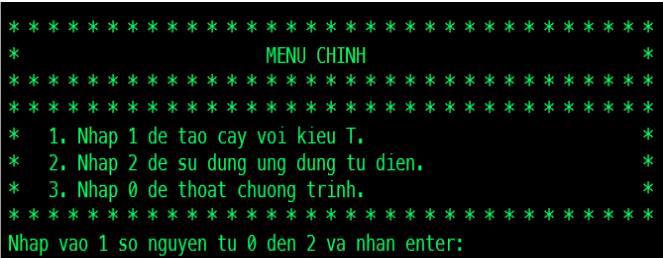

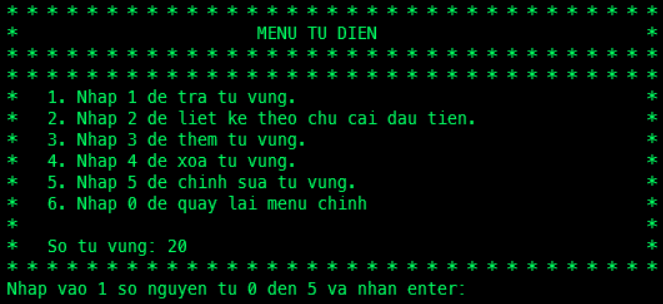
Tất cả các dòng có định dạng gồm 21 ký tự trong đó: 20 ký tự đầu chứa các ký tự của từ nếu không đủ 20 ký tự phải điền vào ký tự NULL cho đến khi đủ 20 ký tự và ký tự 21 là ký tự NULL dùng để kết thúc chuỗi.

- Dòng thứ 1 chứa từ tiếng anh của từ đầu tiên.

- Dòng thứ 2 chứa nghĩa tiếng việt của từ đầu tiên.
- Dòng thứ 3 chứa từ loại của từ đầu tiên.
- Dòng thứ 4 chứa ví dụ của từ đầu tiên.
- Các từ tiếp theo bắt đầu từ dòng kế tiếp từ phía trước và theo định của từ đầu tiên.

4. Thiết kế giao diện

Bảng 12: Danh sách các giao diện chương trình

TT	Màn hình	Mục đích	Giải thích
1	 <pre> ***** * * * MENU CHINH * * * ***** * 1. Nhap 1 de tao cay voi kieu T. * * 2. Nhap 2 de su dung ung dung tu dien.* * 3. Nhap 0 de thoat chuong trinh. * ***** Nhap vao 1 so nguyen tu 0 den 2 va nhan enter: </pre>	Làm giao diện chính cho chương trình	Huỳnh Thanh Tuấn Giúp người dùng có thể lựa chọn thao tác trên cây với kiểu dữ liệu nguyên thủy thông qua lựa chọn 1 và tạo cây với kiểu dữ liệu dữ liệu TuNgu qua lựa chọn 2.
2	 <pre> ***** * * * MENU CAY KIEU T * * * ***** * 1. Nhap 1 de tao cay voi kieu T. * * 2. Nhap 2 de them node vao cay. * * 3. Nhap 3 de tim kiem node trong cay. * * 4. Nhap 4 de xoa node khi biet key. * * 5. Nhap 5 de duyet cay theo thu tu gia/truoc/sau. * 6. Nhap 6 de duyet theo BFS/DFS. * 7. Nhap 7 de xem gia tri nut thu n theo thu tu gia/truoc/sau. * 8. Nhap 8 de xoa nut thu n theo thu tu gia/truoc/sau. * 9. Nhap 9 de them nut vao cay con co goc la nut thu n theo thu tu gia/truoc/sau. * 10. Nhap 0 de xoa cay hien tai va quay tro lai menu chinh. * * Kieu du lieu hien tai: Int So Node hien tai: 0 ***** Nhap vao 1 so nguyen tu 0 den 9 va nhan enter: </pre>	Thao tác cây với kiểu T	Huỳnh Thanh Tuấn Giúp người dùng có thể chọn kiểu dữ liệu cho cây (int, float, char) và thực hiện các thao tác thêm, xoá, tìm kiếm các node và duyệt cây đã tạo.
3	 <pre> ***** * * * MENU TU DIEN * * * ***** * 1. Nhap 1 de tra tu vung. * * 2. Nhap 2 de liet ke theo chu cai dau tien. * 3. Nhap 3 de them tu vung. * * 4. Nhap 4 de xoa tu vung. * * 5. Nhap 5 de chinh sua tu vung. * * 6. Nhap 0 de quay lai menu chinh * * * So tu vung: 20 ***** Nhap vao 1 so nguyen tu 0 den 5 va nhan enter: </pre>	Thao tác với ứng dụng từ điển	Huỳnh Thanh Tuấn Giúp người dùng có thể tra cứu nghĩa tiếng việt, liệt kê theo ký tự và cập nhật (thêm, sửa, xoá) các từ trong từ điển.

IV. Cài đặt và kiểm thử

Bảng 13: Kiểm thử chương trình

Tình huống	Mục đích	Giải thích
Tình huống 1 Chọn chức năng tra từ Dữ liệu vào: nhập số Kết quả dự kiến: chương trình báo lỗi do không nhận số	Kiểm tra dữ liệu nhập từ bàn phím của người dùng.	Kiểm tra dữ liệu đầu vào của người dùng không hợp lệ bằng các nhập số thay chữ trong tình huống người dùng nhập nhầm.
Tình huống 2 Chọn chức năng tra từ Dữ liệu vào: nhập khoảng cách trước và sau từ cần tra Kết quả dự kiến: chương trình vẫn nhận ra từ cần tra	Kiểm tra chương trình khả năng phân biệt từ và khoảng cách.	Bắt tình huống người dùng nhập phím cách khi thực hiện tra cứu từ
Tình huống 3 Chọn chức năng số 1, menu phụ hiện ra, chọn các chức năng khi nhập vào số từ 0 đến 9 Dữ liệu vào: một số lớn hơn 9 hoặc một ký tự không phải số Kết quả dự kiến: chương trình báo lỗi, yêu cầu nhập lại số trong khoảng 0 đến 9	Tránh để người dùng nhập số không có tính năng tương ứng, tránh gây lỗi chương trình.	Bắt tình huống người dùng nhập nhầm nhằm tránh làm cho hệ thống chạy sai khi nhận dữ liệu vào là số khác với các lựa chọn
Tình huống 4 Chọn chức năng số 1, chọn tính năng tạo cây dạng char và sau đó thêm node Dữ liệu vào: một dãy gồm các chữ cái và số lẫn lộn Kết quả dự kiến: chương trình báo lỗi, yêu cầu nhập lại chỉ một chữ cái.	Đảm bảo người dùng nhập vào là một ký tự	Bắt tình huống người dùng nhập vào nhiều hơn 1 ký tự trừ trường hợp ký tự nhiều hơn là phím cách, tab, enter

V. Kết luận

- Các thành viên trong nhóm đã hoàn thành tốt được mục tiêu tạo ra được một chương trình từ điển và thao tác trên cây nhị phân với các kiểu dữ liệu khác nhau.
- **Các khó khăn gặp phải khi thực hiện đề án:** (1) Tìm kiếm cách thao tác trên cây với nhiều kiểu dữ liệu nguyên thủy khác nhau. (2) So sánh các đối tượng được tạo ra từ lớp TuNgu.
- **Cách khắc phục các khó khăn:** (1) Thực hiện tìm kiếm trên internet các đặt tả của vector từ đó tìm được trong c++ có từ khoá template giúp có thể sử dụng nhiều kiểu dữ liệu. (2) Thực hiện nạp chồng lại các toán tử cần thực hiện so sánh khi thao tác với cây.
- **Ý tưởng phát triển đề án:** Từ vựng có bổ sung thêm hình ảnh minh hoạ và cách phát âm của từ. Có thể thao tác sử dụng chuột trên ứng dụng.
- **Ưu điểm:** người dùng có thể tìm từ theo ký tự đầu tiên trong chữ cái. Chính sửa nội dung của từ vựng, thêm, xoá từ vựng
- **Hạn chế:** Khó có thể thực hiện thêm từ vựng tại file data.txt do có ký tự NULL khi ký tự của các từ không đủ 20 ký tự.

VI. Tài liệu tham khảo

1. Chuong-5-Cay Nhi Phan -Tim kiem.pdf
2. Chương 2. Đường đi ngắn nhất.pdf
3. [Ngăn xếp và hàng đợi trong C++](#)
4. [Tại sao gọi cin.clear \(\)](#)
5. [Nạp chồng toán tử trong C++](#)
6. [Template classes trong C++](#)
7. Chapter 15: [C_How to Program_ with an introduction to C++ Global Edition 8th Edition](#)