

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN



ĐỀ TÀI :In bảng tần suất xuất hiện các từ

Người hướng dẫn: THS. NGUYỄN VĂN NGUYỄN

Sinh viên thực hiện:

Bùi Duy Hoàng

LỚP: 21TCLC_KHDL2

Dương Võ Hoàng Hùng

LỚP: 21TCLC_KHDL2

NHÓM :804

Đà Nẵng, 03/2022

MỤC LỤC

Contents

MỤC LỤC	1
DANH MỤC HÌNH VẼ	Error! Bookmark not defined.
MỞ ĐẦU.....	1
1. TỔNG QUAN ĐỀ TÀI.....	Error! Bookmark not defined.
2. CƠ SỞ LÝ THUYẾT	Error! Bookmark not defined.
2.1. Ý tưởng	3
2.2. Cơ sở lý thuyết	3
3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN...Error! Bookmark	
not defined.	
3.1. Phát biểu bài toán.....	4
3.2. Cấu trúc dữ liệu	4
3.3. Thuật toán	4
4. CHƯƠNG TRÌNH VÀ KẾT QUẢ	Error! Bookmark not defined.
4.1. Tổ chức chương trình.....	5
4.2. Ngôn ngữ cài đặt.....	6
4.3. Kết quả	6
4.3.1. Giao diện chính của chương trình	6
4.3.2. Kết quả thực thi của chương trình	7
4.3.3. Nhận xét đánh giá.....	8
5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	9
5.1. Kết luận	9
TÀI LIỆU THAM KHẢO	10
PHỤ LỤC	11

MỞ ĐẦU

Đề tài được thực hiện nhằm mục đích tìm kiếm thông tin trong văn bản 1 cách nhanh hơn ,dễ kiểm soát và tránh nhầm lẫn.Trong một văn bản thường xuất hiện rất nhiều từ dẫn đến việc rất khó tìm kiếm và kiểm soát.Do vậy chúng tôi đã thực hiện chương trình để giải quyết các vấn đề nêu trên.Mục tiêu đề tài là các văn bản chữ cái Latin có độ dài không quá 80 từ và không phân biệt chữ hoa và thường..Phương pháp nghiên cứu trong đồ án là phương pháp quy nạp và duyệt theo thứ tự. Đồ án gồm 5 phần:Tổng quan đề tài,sơ sở lý thuyết,tổ chức cấu trúc dữ liệu và thuật toán,chương trình và kết quả,kết luận và hướng phát triển tương lai.

Chương 1: Tổng quan đề tài

Ta định nghĩa từ là 1 dãy gồm tối đa 10 chữ cái Latinh và trong một văn bản thì mỗi từ được phân cách với các từ kế nó ít nhất 1 ký tự không phải là chữ cái. Ngoài ra, đối với từ không phân biệt giữa chữ thường và chữ in.

Yêu cầu: Viết chương trình bằng ngôn ngữ C thực hiện các công việc sau:

- Đọc file 01 văn bản gồm các dòng có độ dài tối đa 80 , văn bản này được kết thúc bởi một dòng rỗng.
- Đếm và in ra tần suất (số lần xuất hiện) của các từ trong văn bản. Nếu số lượng từ khác nhau trong văn bản nhiều hơn 100 thì chỉ tính tần suất của 100 từ khác nhau xuất hiện đầu tiên.
- In ra bảng tần suất nói trên theo thứ tự ABC.

I.Đầu vào: 01 file văn bản gồm các dòng có độ dài tối đa 80.

I.Đầu ra: 01 file bảng tần suất nói trên theo thứ tự ABC

I.Thuật toán: Sử dụng cấu trúc dữ liệu dạng cây

+ Thực hiện tìm kiếm nút có giá trị x trong cây. Nếu tìm thấy thì trả về vị trí của nút đó , ngược lại chèn nút vào cây nhị phân tìm kiếm.

+ Ta có thể sử dụng giải thuật tìm kiếm và thêm vào cây để sắp xếp thứ tự một dãy: Tạo ra cây nhị phân tìm kiếm, rồi duyệt theo thứ tự LNR

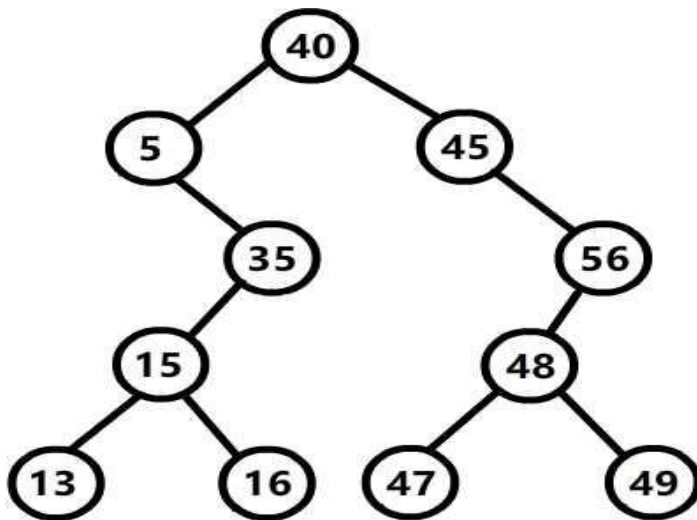
Chương 2: Cơ sở lý thuyết

2.1 Ý tưởng

Để có thể in ra bảng tuần suất xuất hiện các từ, đầu tiên ta phải chuyển các chữ hoa thành thường. Sau đó tách các từ và sử dụng phương pháp để đếm số lần xuất hiện các từ trong văn bản. Để tìm kiếm dễ dàng, ta đưa dữ liệu vào cây nhị phân tìm kiếm rồi duyệt theo kiểu LNR để dễ dàng kiểm soát.

2.2 Cơ sở lý thuyết

Cây nhị phân tìm kiếm là cây nhị phân mà trong đó, các phần tử của cây con bên trái đều nhỏ hơn phần tử hiện hành và các phần tử của cây con bên phải đều lớn hơn phần tử hiện hành.



Hình 1. cây nhị phân tìm kiếm

Nhờ vào đặc tính của cây nhị phân tìm kiếm mà ta có thể dễ dàng kiểm soát và tìm kiếm các phần tử trên cây.

Chương 3:Cấu trúc dữ liệu và thuật toán

3.1 Phát biểu bài toán

- Input:đưa vào 1 file văn bản không phân biệt chữ hoa thường.
- Output:bảng tần suất xuất hiện các từ theo thứ tự anphabet.

3.2 Cấu trúc dữ liệu

- Kiểu kí tự
- Kiểu số nguyên
- Tập nhị phân
- Cây nhị phân tìm kiếm

3.3 Thuật toán

- Bước 1:Tạo file văn bản và mở file
- Bước 2:Dùng hàm lower() chuyển các chữ viết hoa trong văn bản thành chữ thường
- Bước 3:Dùng hàm để tách các từ của văn bản và đưa vào trong mảng 2 chiều
- Bước 4:Dùng hàm để tính số lần xuất hiện các từ đã được tách
- Bước 5:Dùng hàm sortWordList() để sắp xếp các từ đã được tách theo quy tắc anphabet
- Bước 6:Dùng hàm printWordList() để in ra danh sách các từ
- Dùng hàm writeToFile() để kết quả ra file
- Bước 7:Khởi tạo cây nhị phân
- Bước 8:Dùng hàm timkiem() để tìm kiếm các phần tử trong cây nhị phân.
- Bước 9:Nếu tìm kiếm không thấy thì sẽ dùng hàm themphantu() để thêm các phần tử vào trong cây
- Bước 10:Dùng hàm duyet_LNR() để duyệt cây nhị phân tìm kiếm theo kiểu LNR

Chương 4: Chương trình và kết quả

4.1 Tổ chức chương trình

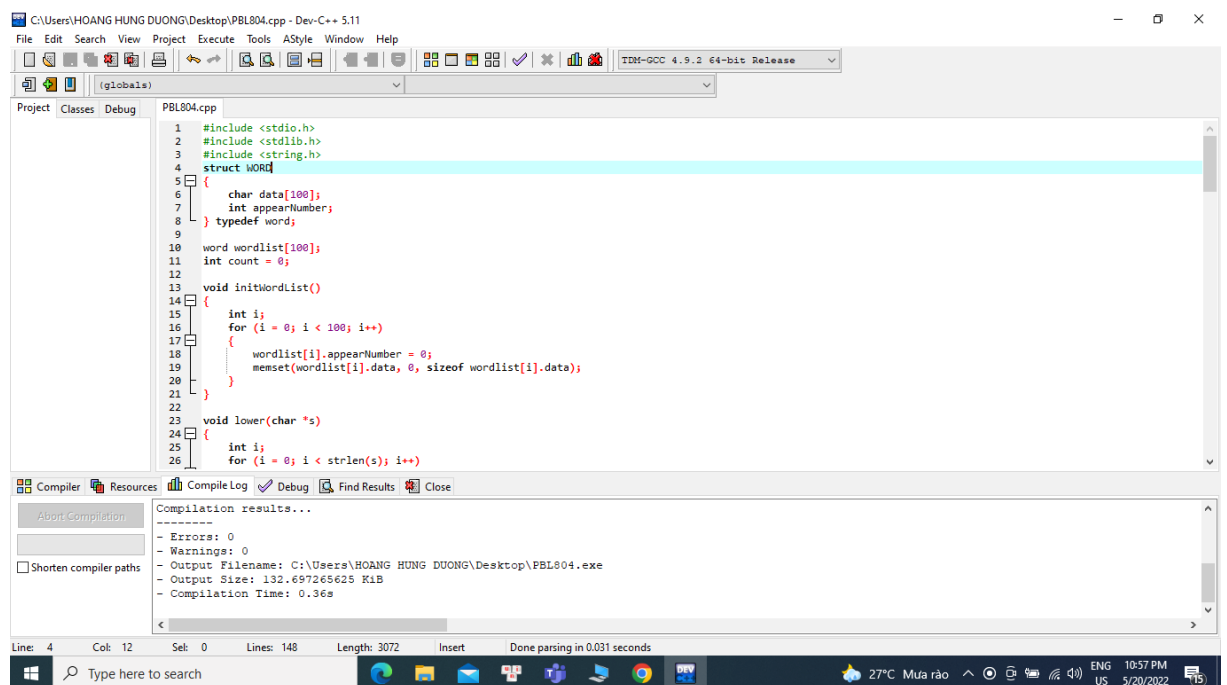
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
Node:struct
NODE* TREE
struct WORD
{
    char data[100];
    int appearNumber;
} typedef word;
word wordlist[100];
int count = 0;
void initWordList()
void lower(char *s)
void addToWordList(word w)
void sortWordList()
void printWordList()
void writeToFile()
Khoitaocay(TREE &t):void
Duyet_LNR(TREE &t):void
Themphantu(TREE &t,int x):void
Timkiem(TREE t):NODE
```

4.2 Ngôn ngữ cài đặt

Ngôn ngữ c

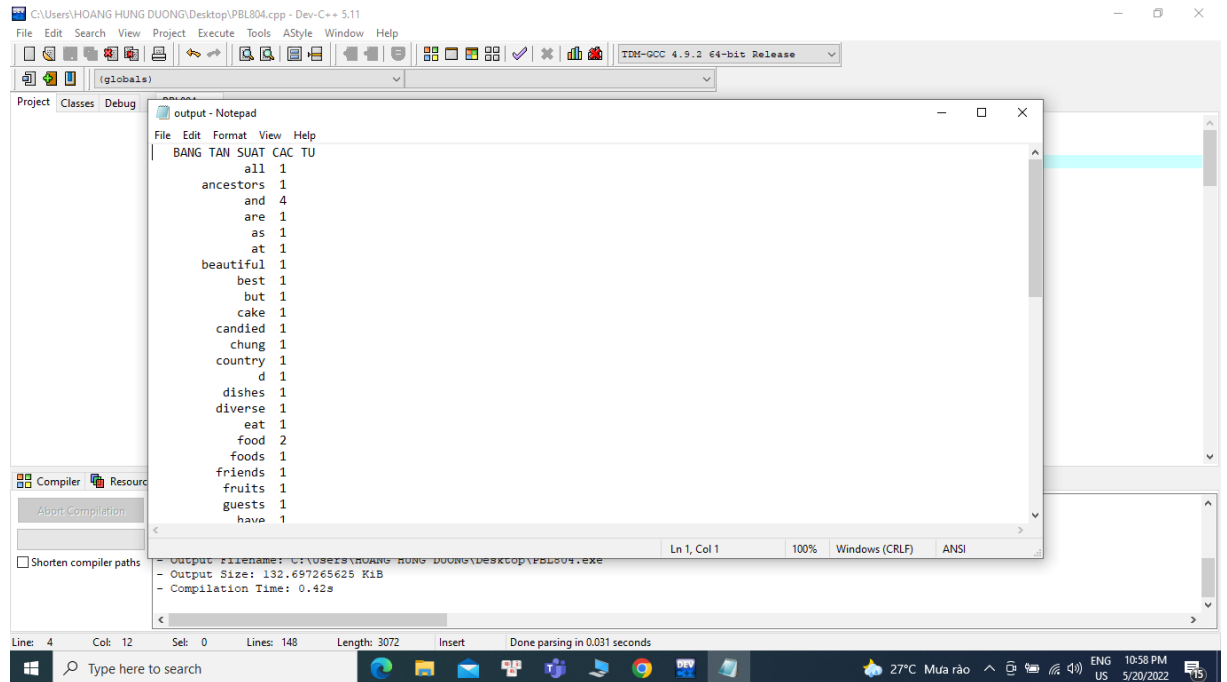
4.3 Kết quả

4.3.1 Giao diện chính của chương trình



Hình2.giao diện chính chương trình

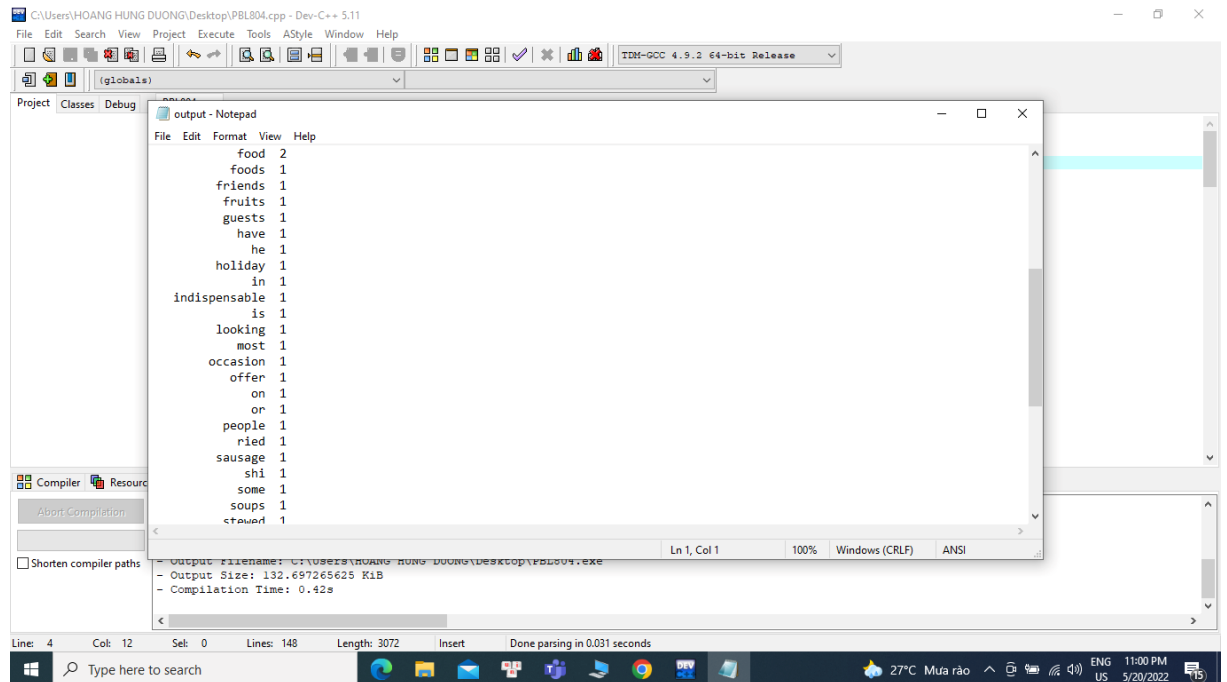
4.3.2 Kết quả thực thi của chương trình



```
File Edit Format View Help
BANG TAN SUAT CAC TU
all 1
ancestors 1
and 4
are 1
as 1
at 1
beautiful 1
best 1
but 1
cake 1
candied 1
chung 1
country 1
d 1
dishes 1
diverse 1
eat 1
food 2
foods 1
friends 1
fruits 1
guests 1
have 1

- Output Filename: C:\Users\HOANG HUNG DUONG\Desktop\pbl804.exe
- Output Size: 132.697265625 KiB
- Compilation Time: 0.42s
```

Hình3.kết quả chương trình



```
File Edit Format View Help
food 2
foods 1
friends 1
fruits 1
guests 1
have 1
he 1
holiday 1
in 1
indispensable 1
is 1
looking 1
most 1
occasion 1
offer 1
on 1
or 1
people 1
ried 1
sausage 1
shi 1
some 1
soups 1
stover 1

- Output Filename: C:\Users\HOANG HUNG DUONG\Desktop\pbl804.exe
- Output Size: 132.697265625 KiB
- Compilation Time: 0.42s
```

Hình4.kết quả chương trình

4.3.3 Nhận xét đánh giá

Chương trình chạy đúng theo yêu cầu. Tuy nhiên cần phải cải thiện thêm vì sử dụng nhiều dữ liệu. Thuật toán còn phức tạp, chưa tối ưu và còn nhiều lỗi. Do vậy dẫn đến việc tiêu tốn thời gian để chạy chương trình.

5.KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1Kết luận

Sử dụng các thuật toán của cây nhị phân tìm kiếm có hiệu quả rất lớn trong việc giải quyết bài toán này so với các thuật toán khác.

5.2Hướng phát triển

Để chương trình hoàn thiện hơn ta cần thêm phần đồ họa vẽ cây nhị phân để dễ dàng hình dung hơn.Cần phát triển để nhận dạng nhiều loại kí tự thay vì chỉ sử dụng kí tự latin.

TÀI LIỆU THAM KHẢO

- [1] *Giáo Trình Kỹ Thuật Lập Trình C Căn Bản Và Nâng Cao* – NXB Bách Khoa Hà Nội.
- [2] Đỗ Xuân Lôi, *Cấu trúc dữ liệu bằng ngôn ngữ C* - NXB Khoa học và kỹ thuật, năm 2003.
- [3] Lê Văn Doanh, Trần Khắc Tuấn, Lê Đình Anh, 101 thuật toán và chương trình chạy bằng ngôn ngữ C, nhà xuất bản khoa học và kỹ thuật, năm 2006.
- [4] https://vnoi.info/wiki/translate/topcoder/dynamic-programming.md?fbclid=IwAR2OOmyIqJUWbk4ALKl63uLeO5nuK_WZZAPhMM8uZx-zght3jWVDgvJRRzE
- [5] https://vi.wikipedia.org/wiki/Quy_ho%E1%BA%A1ch_%C4%91%E1%BB%99ng?fbclid=IwAR2CmTsgiBA92pK1BoWxTgCrRQLA1SbfjgreRA-P0Fe4qSripHuReQtLXhQ
- [6] Một số tài liệu tham khảo khác.

PHỤ LỤC

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <GL/glut.h>
struct WORD
{
    char data[10];
    int appearNumber;
} typedef word;
word wordlist[100];
int count = 0;
void initWordList()
{
    int i;
    for (i = 0; i < 100; i++)
    {
        wordlist[i].appearNumber = 0;
        memset(wordlist[i].data, 0, sizeof wordlist[i].data);
    }
}
void lower(char *s)
{
    int i;
    for (i = 0; i < strlen(s); i++)
    {
```

```
        if (s[i] >= 'A' && s[i] <= 'Z')
        {
            s[i] += 32;
        }
    }
}

void addToWordList(word w)
{
    int i;
    int duplicate = 0;
    for (i = 0; i < count; i++)
    {
        if (strcmp(w.data, wordlist[i].data) == 0)
        {
            wordlist[i].appearNumber++;
            duplicate = 1;
            break;
        }
    }
    if (duplicate == 0)
    {
        w.appearNumber = 1;
        wordlist[count++] = w;
    }
}

void sortWordList()
```

```
{
    int i, j;
    for (i = 0; i < count; i++)
    {
        for (j = i + 1; j < count; j++)
        {
            if (strcmp(wordlist[i].data, wordlist[j].data) > 0)
            {
                word t = wordlist[i];
                wordlist[i] = wordlist[j];
                wordlist[j] = t;
            }
        }
    }
}

void printWordList()
{
    int i;
    for (i = 0; i < count; i++)
    {
        printf("%s: %d\n", wordlist[i].data, wordlist[i].appearNumber);
    }
}

void writeToFile(){
    FILE *file;
    file = fopen("output.txt", "w");
```

```
    int i;
    for ( i = 0; i < count; i++)
    {
        fprintf(file, "%s: %d\n", wordlist[i].data, wordlist[i].appearNumber);
    }
    fclose(file);
}

struct node
{
    int data;
    struct node*pleft;
    struct node*pright;
};

typedef struct node NODE;
typedef NODE* TREE;

//khởi tạo cây
void khoitaocay(TREE &t)
{
    t=NULL;
}

//ham xuất cây nhị phân LNR
void duyet_LNR(TREE &t)
{
    if(t !=NULL)
    {
        duyet_LNR(t->pleft);
```

```
        printf("%d"t->pleft);
        duyet_LNR(t->pright);
    }
}
void themphantu(TREE &t,int x)
{
    if( t==NULL)
    {
        NODE *p = new NODE;
        p->data =x ;
        p->pleft = NULL;
        p->pright = NULL;
        t=p;
    }
    else
    {
        if(x < t->data)
        {
            themphantu(t->pleft,x);
        }
        else if(x> t->data)
        {
            themphantu(t->pright,x);
        }
    }
}
```

```
NODE* timkiem(TREE t)
{
    if(t == NULL)
    {
        return NULL;
    }
    else
    {
        if(x < t->data)
        {
            timkiem(t->pleft,x);
        }
        else if(x > t->pright)
        {
            timkiem(t->pright,x);
        }
        else
        {
            return t;
        }
    }
}

// VE VAY NHI PHAN

void initGL()
{
    glClearColor(0.0f,0.0f,0.0f,1.0f);
}
```

```
        glOrtho(-1,1,-1,1,-1,1);
    }
    void mydisplay()
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0f,0.0f,0.0f);
        glViewport(0.0,600,600);
        glBegin(GL_POLYGON);
            glVertex2f(-0.5f,-0.5f);
            glVertex2f(0.5f,-0.5f);
            glVertex2f(0.5f,0.5f);
            glVertex2f(-0.5f,0.5f);
        glEnd();
        glFlush();
    }
    int main()
    {
        char line[80];
        FILE *file;
        if ((file = fopen("data.txt", "r")) == NULL)
        {
            printf("Error! opening file");
            exit(1);
        }
        initWordList();
        while (fgets(line, sizeof(line), file))
```

```
{
    int i, j = 0;
    word temp;
    for (i = 0; i < strlen(line); i++)
    {
        if ((line[i] >= 'A' && line[i] <= 'Z') || (line[i] >= 'a' && line[i] <=
'z'))
        {
            temp.data[j++] = line[i];
        }
        else
        {
            if (strlen(temp.data) > 0)
            {
                lower(temp.data);
                addToWordList(temp);
                if (count == 100)
                {
                    break;
                }
            }
            memset(temp.data, 0, sizeof temp.data);
            j = 0;
        }
    }
    if (strlen(temp.data) > 0)
```

```
    {  
        lower(temp.data);  
        addToWordList(temp);  
        if (count == 100)  
        {  
            break;  
        }  
    }  
}  
sortWordList();  
printWordList();  
writeToFile();  
fclose(file);  
return 0;  
}
```