**KỸ THUẬT LẬP TRÌNH**

ĐỒ ÁN MÔN HỌC #1  
ĐẾM TỪ TIẾNG VIỆT

Tên SV : Phạm Minh Chiến

MSSV : 1612052  
Môn : Kỹ thuật lập trình

Tên đồ án : **Đếm từ Tiếng việt**

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Khoa Công nghệ Thông tin

Đại học Khoa học Tự nhiên TP HCM

Mục lục

[1. Mô tả đồ án 3](#_Toc485293438)

[2. Xử lí tiếng việt trong C/C++: 3](#_Toc485293439)

[3. Thuật toán 3](#_Toc485293440)

[3.1. Thuật toán phân tách từ: 3](#_Toc485293441)

[3.2. Tìm từ đơn: 4](#_Toc485293442)

[3.3. Đếm từ ghép (cơ bản) 5](#_Toc485293443)

[3.4. Đếm từ ghép nâng cao 5](#_Toc485293444)

[3.5. Đếm từ láy 5](#_Toc485293445)

[3.6. Đếm từ có nghĩa 6](#_Toc485293446)

[4. Hướng dẫn sử dụng: 7](#_Toc485293447)

# Mô tả đồ án

* Đầu vào: một câu hoặc một đoạn văn bản tiếng việt có nghĩa.
* Đầu ra: đếm số lượng các từ loại  
  + Từ đơn

+ Từ ghép  
+ Từ láy  
+ Từ

# Xử lí tiếng việt trong C/C++:

Sử dụng kiểu dữ liệu mở rộng của C++ là wchar\_t trong thư viện <wchar.h> để lưu trữ tiếng việt với kiểu encode UTF-8 hoặc UTF-16. Kiểu dữ liệu wchar\_t cũng tương tự kiểu char về mặt ý nghĩa, chỉ có khác các hàm xử lí (trong wchar.h) là khác và kiểu xuất cũng khác, phải chuyển qua dòng nhập xuất stdin, stdout sang kiểu \_O\_U8TEXT (mặc định là O\_TEXT chứa trong thư viện <fcntl.h>) bằng hàm setmode có trong thư viện <io.h>:

\_setmode(\_fileno(stdout), \_O\_U8TEXT);

\_setmode(\_fileno(stdin), \_O\_U8TEXT);

Như vậy bây giờ ta sử dụng chuỗi wchar\_t tương tự như char.

# Thuật toán

* 1. **Thuật toán phân tách từ:**

Với dữ liệu đầu vào là một đoạn văn bản tiếng việt có dấu thì việc phân việt các từ (word) dựa vào dấu khoảng trắng. Ý tưởng để xét tất cả các trường hợp từ có nghĩa trong văn bản là coi mỗi “từ” (phân biệt bởi dấu khoảng trắng), giả sử đoạn có N từ thì coi đoạn đó là một mảng A có N phần tử. Từ mảng A ta chọn ra tất cả các mảng con liên tiếp có nhiều nhất 4 phần tử lưu trữ vào mảng Data làm dữ liệu để xử lí. Như vậy ta có mảng Data là mảng dữ liệu cơ sở chứa tất cả các từ có thể có nghĩa trong tiếng việt.

Ví dụ: Đầu vào: “Thuật toán tách từ tiếng việt”

* Mảng A có N = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **N** | **0** | **1** | **2** | **3** | **4** | **5** |
| **A** | thuật | toán | tách | từ | tiếng | viêt |

* Mảng Data

|  |  |  |
| --- | --- | --- |
| **Data** | **Chọn** | **Nội dung** |
| 0 | A[0] | thuật |
| 1 | A[1] | toán |
| 2 | A[2] | tách |
| 3 | A[3] | từ |
| 4 | A[4] | tiếng |
| 5 | A[5] | việt |
| 6 | A[0], A[1] | thuật toán |
| 7 | A[1], A[2] | toán tách |
| 8 | A[2], A[3] | tách từ |
| 9 | A[3], A[4] | từ tiếng |
| 10 | A[4], A[5] | Tiếng việt |
| 11 | A[1], A[2], A[3] | thuật toán tách |
| 12 | A[2], A[3], A[4] | toán tách từ |
| 13 | A[3], A[4], A[5] | từ tiếng việt |
| 14 | A[1], A[2], A[3], A[4] | thuật toán tách từ |
| 15 | A[2], A[3], A[4], A[5] | toán từ tiếng việt |

Từ mảng Data ta có thể tìm được các từ loại theo yêu cầu:

* 1. **Tìm từ đơn:**

Từ đơn trong tiếng việt là từ được cấu tạo từ tiếng có nghĩa. Việc xét xem từ cần kiểm tra có nghĩa hay không tương đương việc từ đó có xuất hiện trong từ điển hay không.

* Do đó thuật toán đếm từ đơn trong Data:

For (i=0->size\_of\_\_data)  
 if (*kiemTraCoNghia (***data[i]**) and *demSoTu* (**data[i]**)==1)  
 **count\_Tu\_Don++**  
 output(**data[i]**)

End if

End for

Trong đó có 2 hàm xuất hiện:   
+ bool kiemTraCoNghia(wchar\_t\*s): kiểm tra chuỗi s có xuất hiện trong file từ điển hay không. Trả về True nế có và Flase nếu không. (không phân biệt hoa thường)

+ int demSoTu(wchar\_t\*s); đếm số từ (ngăn cách bởi khoảng trắng) trong chuỗi s.

* 1. **Đếm từ ghép (cơ bản)**

Từ ghép là từ có 2 tiếng trở lên và có nghĩa. Do đó việc kiểm tra data[i] có phải là từ ghép hay không tương đương kiểm tra demSoTu(data[i])>=2 và kiemTraCoNghia(data[i]) ==true.

Thuật toán kiểm tra từ ghép:

For (i=0->size\_of\_\_data)  
 if (*kiemTraCoNghia (***data[i]**) and *demSoTu* (**data[i]**)>1)  
 **count\_Tu\_Ghep++**  
 output(**data[i]**)

End if

End for

* 1. **Đếm từ ghép nâng cao**

Về thuật toán sẽ trình bày trong mục 3.6.

* 1. **Đếm từ láy**

Trên lý thuyết tiếng việt thì việc nhận dạng từ láy hoàn toàn phụ thuôc và ngữ điệu của từ mà phân biệt thành láy âm, lấy vần… Chung quy từ láy có tính chất như sau:

Giả sử ta có một từ đơn có nghĩa là: AB. Trong đó:  
+ A được coi là thành phần gồm các chữ cái là **phụ âm** A đứng độc lập thì không tạo thành từ đơn có nghĩa (ví dụ: b, c, d, đ, p, ph, ch, l, nh, ng, ngh,...), A có thể là rỗng.  
+ B: đặt B=B1B2 với B1 là tập các chữ cái là **nguyên âm** kết hợp với nhau, B2 là tập có cấy trúc tương tự A. B1B2 khi đứng độc lập vẫn có nghĩa (về ngư pháp). B phải khác rỗng. *Ví dụ: “chiến”: A={c, h}, B={i, ế, n}, <B1={i, ế}, B2={n}>*

* Khi đó AB CD (từ ghép) là một từ láy nếu thỏa: A==C hay B==D
* Tổng quát:

***kiemTraTuLay***(“A1B1 A2B2 … AnBn”) ==TRUE ⬄

,với n>1

Từ đó ta có thuật toán Đếm từ láy:

For (i=0->size\_of\_\_data)  
 if (*kiemTraCoNghia (***data[i]**) and *kiemTraTuLay* (**data[i]**))  
 **count\_Tu\_Lay++**  
 output(**data[i]**)

End if

End for

* 1. **Đếm từ có nghĩa**

Vấn phân tách các từ trong đoạn văn thành các từ, cụm từ có nghĩa (khác với việc kiểm tra có nghĩa tồn tại trong từ điển như trên) đang là đề tài nghiên cứu của nhiều nhóm nghiên cứu về xử lí ngôn ngữ tự nhiên (tiếng việt). Xem xét từ có nghĩa trong đoạn văn liên quan đến việc xem xét ngữ cảnh, tần suất xuất hiện của từ trong một kho dữ liệu cho trước (thuật toán TF-IDF, HashingTF, Vectorizal),… Do nó nghe lớn lao quá nên em xin đề xuất một thuật toán như sau (đúng được vài trường hợp):

Xét một câu chuẩn về mặt ngữ pháp: (trang ngữ) + chủ ngữ + động từ + danh từ. Không bao gồm các hư từ (thuật ngữ tiếng anh gọi là hư stopwords) là các từ chỉ trạng thái, vị trí tương đối như: “ơ, trên, dưới, tại, trong, ngoài, …”, từ chỉ số lượng, và các từ mang nghĩa chung chung được ghép với các danh từ khác để tạo từ có nghĩa. Thay các hư từ thành các động từ có nghĩa. Ví dụ thay vì câu ”Hôm qua, em ở trên máy bay” thì hãy dùng là “Hôm qua, em đi máy bay”. Cố gắng không dùng các stopwords nếu có thể thay thế.   
Ta tiến hành đọc từ đầu câu trở đi lần lượt đọc các cụm trạng ngữ -> chủ ngữ -> động từ -> danh từ. Đọc chủ ngữ xong thì việc đọc động từ sẽ không bao gồm phần chứa chủ ngữ trở về trước. Nếu có một chuỗi không thuộc phần nào thì ta bỏ qua (trường hợp stopword).

Cải tiến: liệt kê tất cả các stopword vào một file tên stopwords\_file.txt, mỗi lần xét một chuỗi thì kiểm tra nếu chuỗi đó không có nghĩa trong từ điển thì xem từ đầu tiên trong chuỗi có phải là một stopword không, nếu có thì ta xét tiếp” chuỗi mới = chuỗi cũ – stopword”.

* Thuật toán:

For (k=4 -> 1)

For (i=0->size\_of\_\_data)

If(demSoTu(data[i]==k)  
 if (*kiemTraCoNghia (***data[i]**))  
 **count\_Tu\_++**  
 output(**data[i]**)

Else

viTriTach = LayKTuDauTien(data[i], Temp\_stw, 1);

Temp\_S=Data[i]+viTriTach

If (kiemTraStopWords(Temp\_swt) và kiemTraCoNghia(S))  
 count\_Tu++;

Break;

End if

End else

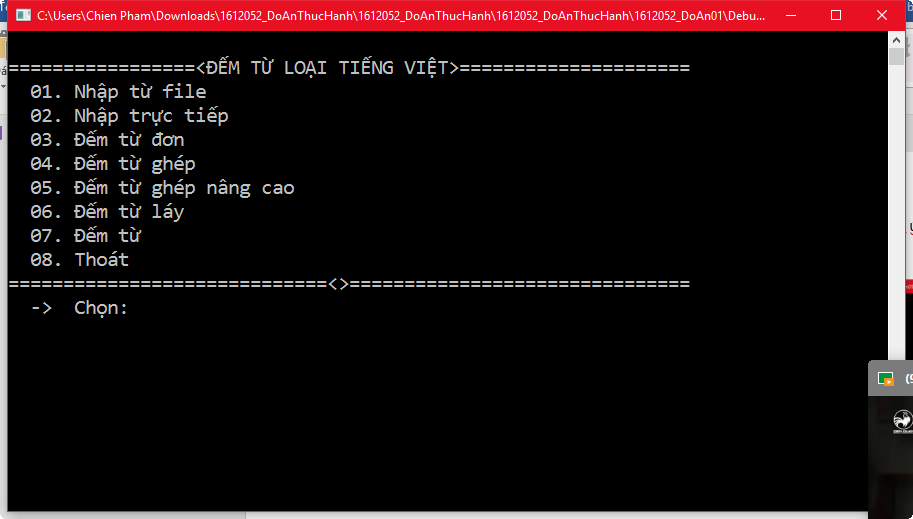
End if

End if

End for

End for

# Hướng dẫn sử dụng:

Chương trình tổ chức kiểu Menu bảng chọn, đọc các lựa chọn và nhập số tương ứng:

**01. Nhập từ file:**

- Đọc chuỗi từ file “test\_File.txt

**02. Nhập trực tiếp:**

- Nhập chuỗi từ bàn phím

**03. Đếm từ đơn**

- Đếm, liệt kê số từ đơn có trong câu

**04. Đếm từ ghép:**

- Đếm, liệt kê số từ ghép (cơ bản)

**05. Đếm từ ghép nâng cao:**

- Đếm, liệt kê số từ ghép (nâng cao)

**06. Đếm từ láy**

- Đếm số lượng từ láy trong câu

- Liệt kê các từ láy có trong câu

**07. Đếm từ**

- Đếm, liệt kê số lượng từ có nghĩa trong câu (nâng cao)

**08. Thoát**