BÁO CÁO BÀI TẬP 4

**Môn học: Lập trình hướng đối tượng**

**Tên chủ đề: báo cáo lab04**

*GVHD: Nguyễn Hữu Quyền*

*Ngày báo cáo: 22/05/2023*

**Nhóm: 02 (nếu không có xoá phần này)**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: IT002.N28.2

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Hải Phong | 22521088 | 22521088@gm.uit.edu.vn |
| 2 | Hồ Trung Kiên | 22520704 | 22520704@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** | **Người đóng góp** |
| 1 | Câu hỏi 01 | 100% | Nguyễn Hải Phong |
| 2 | Câu hỏi 02 | 100% | Nguyễn Hải Phong  Hồ Trung Kiên |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

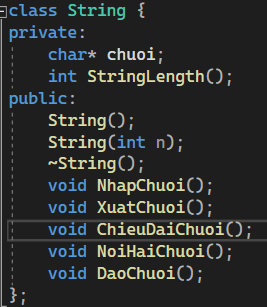
BÁO CÁO CHI TIẾT BÀI THỰC HÀNH

## Câu hỏi 01

* + - Tài nguyên: Lớp String có thành phần dữ liệu là chuỗi ký tự char \* và các hàm nhập, xuất chuỗi và các hàm cần thiết theo yêu cầu đề bài.
* Mô tả/mục tiêu: Viết định nghĩa lớp String để biểu diễn khái niệm chuỗi ký tự với các phương thức thiết lập và huỷ bỏ, các hàm thành phần tính chiều dài chuỗi, nối hai chuỗi, đảo chuỗi, nhập và xuất chuỗi.
* Các bước thực hiện/Phương pháp thực hiện:
  + - * Bước 1: Khai báo các thư viện cần thiết gồm các thư viện iostream (dùng cho việc nhập xuất) và sử dụng using namespace std để tối ưu hóa cho đoạn code.



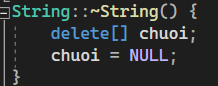
* Bước 2: Khởi tạo lớp String có thành phần dữ liệu trong private là chuỗi ký tự có dạng là char \* (con trỏ, trỏ đến vùng nhớ có kiểu dữ liệu là char), hàm StringLength để tính chiểu dài chuỗi có kiểu trả về là int (dùng để phục vụ trong các hàm phương thức trong public) và trong public gồm các hàm thiết lập và hàm hủy chuỗi, các phương thức nhập, xuất chuỗi ký tự, hàm tính chiều dài chuỗi, hàm nối hai chuỗi cũng như hàm đảo chuỗi.



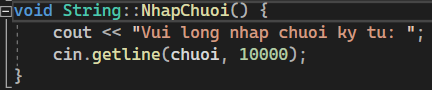
Ở hàm khởi tạo (constructor), ta sẽ cho cấp phát động cho chuỗi ký tự bằng cách dùng lệnh new char, đồng thời ta có thể cấp phát cho chuỗi ký tự bao nhiêu cũng được (ở đây thì mình cấp phát động cho chuỗi là 10.000 ký tự)



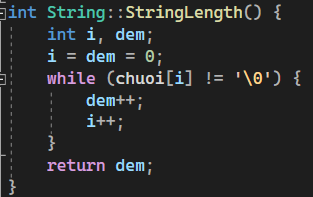
Còn ở hàm hủy (destructor), thì ta sẽ tiến hành dùng lệnh delete[] để xóa mảng được cấp phát động trong hàm constructor và đồng thời ta sẽ trỏ con trỏ đến NULL để tránh việc con trỏ sau khi xóa sẽ trỏ đến vùng không mong muốn.



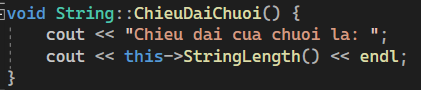
Sau đó là hàm nhập hay còn gọi là hàm cập nhật. Hàm này sẽ cho phép người dùng nhập vào chuỗi ký tự bất kỳ cho đến khi người dùng nhấn Enter thì dừng với giới hạn chuỗi ký tự tối đa là 10.000 ký tự bằng hàm cin.getline. Nếu như chuỗi ký tự mà vượt quá giới hạn đó thì hàm chỉ lấy đúng 10.000 ký tự và bỏ qua những ký tự sau đó.



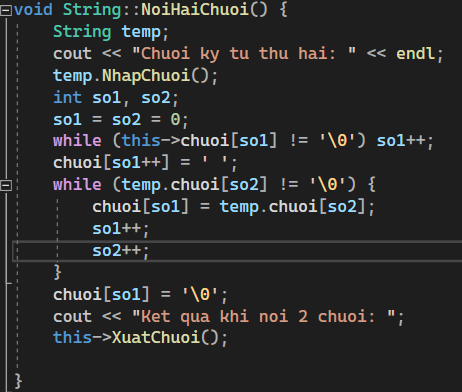
Hàm dùng để tính chiều dài của chuỗi (hàm StringLength có kiểu trả về là int) sẽ sử dụng một vòng lặp while để kiểm tra từng ký tự có trong chuỗi. Ta sẽ bằng đầu kiểm tra từ đầu chuỗi, tức là vị trí bằng 0, nếu như ký tự đó khộng phải là null byte (‘\0’ dùng để kết thúc chuỗi) thì ta sẽ đếm đó là 1 ký tự và liên tục như vậy cho đến khi gặp ký tự kết thúc chuỗi và trả về kết quả của vòng lặp đó.



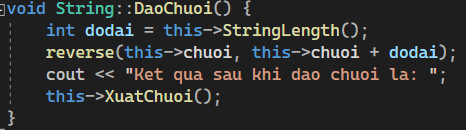
Hàm phương thức tính chiều dài của chuỗi trong phần public sẽ gọi đến hàm StringLength đã được khai báo ở trên để tính chiều dài của chuỗi được nhập vào.

****

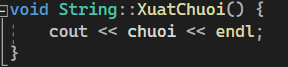
Hàm nối 2 chuỗi thì ta sẽ khai báo thêm một biến có kiểu dữ liệu là lớp String và gọi đến hàm nhập chuỗi để yêu cầu người dùng nhập vào chuỗi thứ hai để tiến hành nối chuỗi. Sau đó ta sẽ chạy một vòng lặp while cho chuỗi đầu tiên để tìm chiều dài của chuỗi đầu tiên bằng cách kiểm tra từng ký tự cho là null byte không. Sau đó ta chèn thêm dấu khoảng trắng vào cuối chuỗi 1. Sau đó thêm một vòng lặp while thứ hai để chạy từ đầu đến cuối chuỗi 2 và chèn từng ký tự của chuỗi 2 vào cuối chuỗi thứ nhất và cho đến khi chuỗi thứ hai gặp ký tự null byte thì dừng. Cuối cùng ta không quên chèn vào cuối chuỗi 1 sau khi được nối 2 chuỗi ký tự null để khi xuất ra chuỗi thì gặp đến null byte thì sẽ tự động dừng lại, không đọc những gì phía sau null byte. Đồng thời ta sẽ gọi đến hàm xuất để xuất ra chuỗi đó.



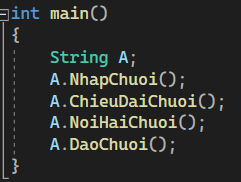
Hàm đảo chuỗi thì đầu tiên ta gọi đến hàm StringLength để tính chiều dài của chuỗi. Sau đó ta sử dụng lệnh reverse trong đó ta cần truyền vào vị trí đầu chuỗi và vị trí cuối của chuỗi. Vì đây là mảng các ký tự nên mặc định khi truyền vào thông số đầu tiên là chính bản thân chuỗi thì chương trình sẽ hiều đó là vị trí đầu tiên của chuỗi, còn thông số thứ hai là chuỗi + chiều dài chuỗi thì chương trình hiểu đó là vị trí cuối cùng trong chuỗi. Sau đó ta sẽ gọi đến hàm xuất để xuất ra kết quả chuỗi được đảo ngược.



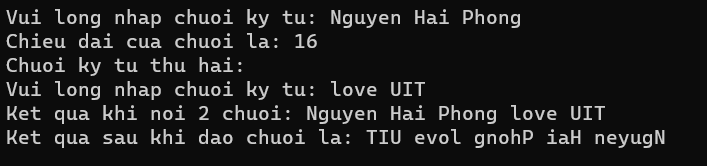
Cuối cùng là hàm xuất dùng để xuất ra chuỗi thì ta chỉ việc đơn giản gọi đến lệnh cout để xuất ra chuỗi.



* Bước 3: Trong hàm main thì ta lần lượt khai báo biến A có kiểu dữ liệu là lớp String. Và đồng thời gọi đến hàm nhập để nhập chuỗi, hàm tính chiều dài chuỗi, hàm nỗi hai chuỗi và hàm đảo ngược chuỗi.



Ví dụ demo:



## Câu hỏi 02:

* + - Tài nguyên: lớp string và vector trong thư viện STL
* Mô tả/mục tiêu: Áp dụng các lớp string và vector trong thư viện STL để xây dựng chương trình thống kê đoạn văn như sau:

- Đọc một đoạn văn từ file văn bản.

- Đếm số lượng câu trong đoạn văn (câu kết thúc thúc bởi dấu ., !, ?).

- Đếm số lượng từ trong mỗi câu

- Tìm từ xuất hiện nhiều nhất trong đoạn văn (có thể có nhiều từ).

- Sắp xếp tăng dần các từ (theo thứ tự từ điển) trong mỗi câu.

- Kết xuất kết quả ra file văn bản theo cấu trúc sau:

• Dòng đầu tiên chứa C là số lượng câu.

• C dòng tiếp theo chứa Ti là số lượng từ trong câu thứ i (1 <= i <= C).

• Dòng kế tiếp chứa các từ xuất hiện nhiều nhất.

• Các dòng kế tiếp chứa đoạn văn sau khi đã sắp xếp

* Các bước thực hiện/Phương pháp thực hiện:

A screen shot of a computer program

Description automatically generated with low confidence

* + - * Bước 1: Khai báo các thư viện và namespace:
* #include <iostream>: Để sử dụng các chức năng nhập/xuất trong C++.
* #include <fstream>: Để làm việc với các tệp tin.
* #include <vector>: Để sử dụng vector, một cấu trúc dữ liệu dạng mảng động.
* #include <string>: Để sử dụng chuỗi ký tự.
* #include <sstream>: Để thao tác với chuỗi ký tự dạng luồng (stringstream).
* #include <algorithm>: Để sử dụng các chức năng xử lý dữ liệu trong STL (Standard Template Library).
* #include <map>: Để sử dụng cấu trúc dữ liệu map trong STL.
* using namespace std;: Đặt trong namespace std để không cần ghi rõ std:: trước các tên thư viện của C++.

Bước 2: Khai báo và khởi tạo các biến toàn cục:



* string DauCuoiCau = "!.?";: Chuỗi các dấu chấm câu cuối câu.
* string DauKhongLaTu = ",;:()[]{}/\\\"";: Chuỗi các dấu câu không phải là từ.

A picture containing text, screenshot, software, font

Description automatically generated

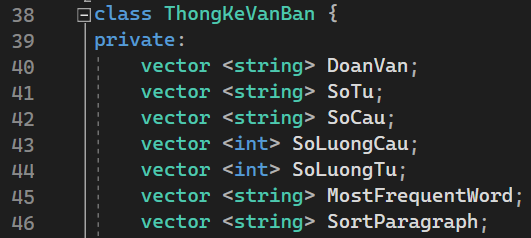
* vector<string> tachchu(const string& str, const string delimiter): Hàm để tách chuỗi thành các từ dựa trên delimiter.

A picture containing text, screenshot, font

Description automatically generated

* string thaythe(string source, string old\_str, string new\_str): Hàm để thay thế tất cả các xuất hiện của old\_str bằng new\_str trong chuỗi source.
* Bước 3: Khai báo và khởi tạo lớp ThongKeVanBan:

Đây là một lớp đối tượng được sử dụng để thực hiện các phương thức thống kê văn bản. Các thành phần dữ liệu bao gồm:



* vector<string> DoanVan: Mảng chứa các đoạn văn bản.
* vector<string> SoTu: Mảng chứa các từ trong văn bản.
* vector<string> SoCau: Mảng chứa các câu trong văn bản.
* vector<int> SoLuongCau: Mảng chứa số lượng câu trong từng đoạn văn.
* vector<int> SoLuongTu: Mảng chứa số lượng từ trong từng câu.
* vector<string> MostFrequentWord: Mảng chứa các từ xuất hiện nhiều nhất trong văn bản.
* vector<string> SortParagraph: Mảng chứa các đoạn văn đã được sắp xếp.

Các phương thức bao gồm:

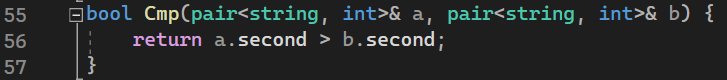
A screen shot of a computer program

Description automatically generated with low confidence

* ThongKeVanBan(): Hàm khởi tạo của lớp, đọc và xử lý văn bản từ tệp tin đầu vào.
* void DocSoLuongCau(): Phương thức để đọc số lượng câu trong từng đoạn văn.
* void DocSoLuongTuTrongCau(): Phương thức để đếm số lượng từ trong mỗi câu.
* void TuXuatHienNhieuNhat(): Phương thức để tìm các từ xuất hiện nhiều nhất trong văn bản.
* void SapXepDoanVan(): Phương thức để sắp xếp các đoạn văn theo thứ tự từ điển.
* void XuatFileVanBan(): Phương thức để xuất thông tin thống kê vào tệp tin kết quả.

Đến với chi tiết cách hoạt động của từng phương thức trong lớp ThongKeVanBan:

Đầu tiên ta khởi tạo hàm bool Cmp(pair<string, int>& a, pair<string, int>& b): Hàm so sánh để sắp xếp các cặp (từ, tần số) theo thứ tự giảm dần của tần số xuất hiện.



1. Phương thức ThongKeVanBan() (constructor):

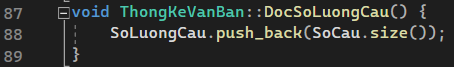
A picture containing text, screenshot, software, multimedia software

Description automatically generated

* Thực hiện việc đọc và xử lý văn bản từ tệp tin đầu vào ("test.txt").
* Trong phương thức này, các bước sau được thực hiện:
  + Tách đoạn văn thành các câu và lưu vào mảng DoanVan.
  + Tách các câu thành các từ và lưu vào mảng SoTu..

1. Phương thức DocSoLuongCau():

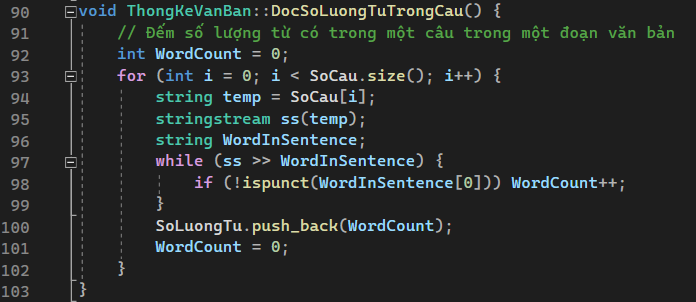
Hàm này được sử dụng để đọc số lượng câu trong đoạn văn và lưu vào mảng SoLuongCau.



* Đầu tiên, phương thức sử dụng phương thức size() để lấy số lượng câu trong mảng SoCau, và sau đó, sử dụng phương thức push\_back() để thêm số lượng câu này vào mảng SoLuongCau.
* Kết quả là số lượng câu được đọc từ đoạn văn và lưu vào mảng SoLuongCau.

1. Phương thức DocSoLuongTuTrongCau():

Hàm này được sử dụng để đếm số lượng từ trong mỗi câu trong đoạn văn và lưu vào mảng SoLuongTu.



* Trong phương thức, ta khởi tạo một biến WordCount để đếm số từ trong mỗi câu. Ban đầu, biến này được đặt bằng 0.
* Sau đó, phương thức duyệt qua từng câu trong mảng SoCau bằng vòng lặp for.
* Đối với mỗi câu, ta tạo một stringstream từ câu đó để tách từ. Sau đó, ta sử dụng vòng lặp while để đọc từng từ trong câu.
* Trong vòng lặp, ta kiểm tra xem từ hiện tại có bắt đầu bằng dấu câu hay không, bằng cách sử dụng hàm ispunct() để kiểm tra ký tự đầu tiên của từ. Nếu không phải dấu câu, ta tăng biến WordCount lên 1.
* Sau khi đọc hết câu, ta thêm giá trị WordCount (số lượng từ trong câu đó) vào mảng SoLuongTu bằng cách sử dụng phương thức push\_back().
* Cuối cùng, ta đặt lại biến WordCount về 0 để chuẩn bị đếm số từ cho câu tiếp theo.
* Kết quả là mảng SoLuongTu sẽ chứa số lượng từ tương ứng với từng câu trong đoạn văn.

1. Phương thức TuXuatHienNhieuNhat ():

Hàm TuXuatHienNhieuNhat() được sử dụng để tìm từ xuất hiện nhiều nhất trong đoạn văn.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

* Đầu tiên, ta khởi tạo một vector WordInParagraph để chứa các từ trong đoạn văn.
* Tiếp theo, ta hợp các từ trong mảng SoTu thành một đoạn văn bằng cách ghép các từ lại với nhau và cách nhau bởi dấu cách.
* Sau đó, ta tạo một stringstream từ đoạn văn để tách các từ. Trong vòng lặp, ta tách từng từ và biến chúng thành chữ thường (để xử lý trường hợp từ viết hoa và từ viết thường được coi là giống nhau).
* Các từ sau khi đã được xử lý được đưa vào vector WordInParagraph.

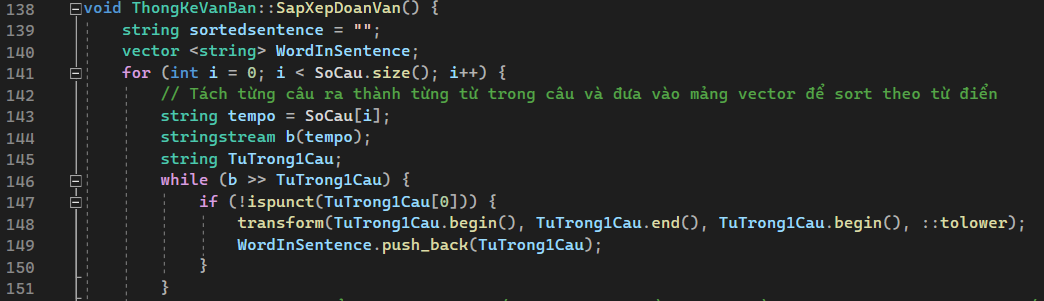
A picture containing text, screenshot, software, multimedia software

Description automatically generated

* Tiếp theo, ta tạo một map A để đếm số lần xuất hiện của từng từ trong WordInParagraph. Key của map là từ, và value là số lần xuất hiện.
* Sau đó, ta tạo một vector B và sao chép các cặp key-value từ map A vào vector B.
* Sử dụng hàm sort() để sắp xếp vector B theo thứ tự giảm dần của số lần xuất hiện.
* Ta lấy số lần xuất hiện nhiều nhất của từ và từ đó gán cho biến Max và từ đó gán cho biến t.
* Cuối cùng, ta duyệt qua các cặp key-value trong vector B, nếu có từ nào có số lần xuất hiện như Max, ta thêm từ đó vào biến t. Nếu có từ khác có số lần xuất hiện khác Max, ta đưa từ t vào mảng MostFrequentWord và kết thúc vòng lặp.
* Cuối cùng, ta xóa mảng WordInParagraph và các biến A và B để giải phóng bộ nhớ.

1. Phương thức SapXepDoanVan():

Hàm SapXepDoanVan() của lớp ThongKeVanBan được sử dụng để sắp xếp các câu trong đoạn văn theo thứ tự từ điển và viết hoa chữ cái đầu của từ đầu tiên trong mỗi câu.



* Đầu tiên, ta khởi tạo một chuỗi sortedsentence để chứa đoạn văn đã được sắp xếp.
* Tiếp theo, ta duyệt qua mảng SoCau chứa các câu trong đoạn văn.
* Trong vòng lặp, ta lấy câu thứ i từ mảng SoCau và tạo một stringstream b để tách từ trong câu.
* Ta tách từng từ trong câu và biến chúng thành chữ thường, sau đó đưa vào mảng WordInSentence.

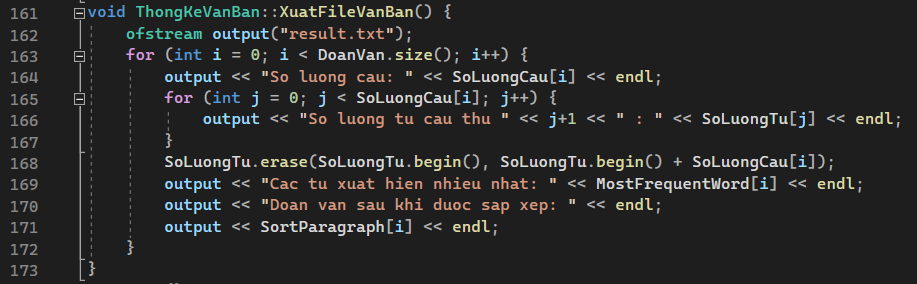
A picture containing text, screenshot, font

Description automatically generated

* Sau đó, ta sắp xếp các từ trong câu theo thứ tự từ điển bằng cách sử dụng hàm sort().
* Ta viết hoa chữ cái đầu của từ đầu tiên trong câu bằng cách sử dụng hàm toupper().
* Cuối cùng, ta thêm dấu chấm vào cuối câu và ghép các từ lại thành câu đã sắp xếp bằng cách sử dụng vòng lặp và chuỗi sortedsentence.
* Sau khi duyệt qua tất cả các câu trong đoạn văn, ta đưa đoạn văn đã sắp xếp vào mảng SortParagraph.
* Mảng WordInSentence được xóa để giải phóng bộ nhớ.

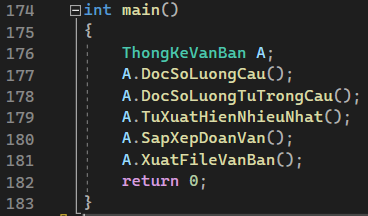
1. Phương thức SapXepDoanVan():

Phương thức này sẽ ghi thông tin thống kê về đoạn văn vào tệp "result.txt" để lưu trữ hoặc sử dụng cho mục đích khác.



* Đầu tiên, ta tạo một đối tượng ofstream với tên output để ghi dữ liệu vào tệp văn bản có tên "result.txt".
* Tiếp theo, ta duyệt qua mảng DoanVan để xuất thông tin về từng đoạn văn.
* Trong vòng lặp đầu tiên, ta xuất số lượng câu trong đoạn văn bằng cách sử dụng SoLuongCau[i].
* Tiếp theo, ta duyệt qua từng câu trong đoạn văn và xuất số lượng từ trong mỗi câu bằng cách sử dụng SoLuongTu[j].
* Sau khi xuất số lượng từ trong từng câu, ta xóa các phần tử đã xuất khỏi mảng SoLuongTu bằng cách sử dụng hàm erase().
* Tiếp theo, ta xuất các từ xuất hiện nhiều nhất trong đoạn văn bằng cách sử dụng MostFrequentWord[i].
* Sau đó, ta xuất thông tin về đoạn văn đã được sắp xếp bằng cách sử dụng SortParagraph[i].
* Cuối cùng, tệp văn bản được đóng tự động sau khi chạy xong.

Bước 4: Khởi tạo hàm main():



* Tạo biến “A” thuộc lớp ThongKeVanBan.
* Tiến hành phân tích văn bản “A” với các hàm:
  + A.DocSoLuongCau()
  + A.DocSoLuongTuTrongCau()
  + A.TuXuatHienNhieuNhat()
  + A.SapXepDoanVan()
* Cuối cùng xuất ra file văn bản với hàm A.XuatFileVanBan() và kết thúc chương trình.

VD demo:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)