

BÀI 1 (BÀI SỐ 7 TRONG DANH SÁCH BÀI TẬP)

**1.1. Đề bài**

Bài 1: Xây dựng lớp biểu diễn một bảng băm sử dụng một trong những hàm băm đã học o Sử dụng lớp bảng băm để xây dựng lớp biểu diễn từ điển Anh Việt. Mỗi phần tử của từ điển là một cặp (tiếng anh, nghĩa tiếng việt). Với các phương thức:

a. Nạp từ điển từ file vào bảng băm

b. Tìm kiếm bằng phương pháp tìm kiếm trên bảng băm

c. Sửa đổi các từ

d. Thêm từ mới

e. Xóa từ

f. Lưu từ điển vào file

Bài 2: Xây dựng chương trình có các chức năng

a. Nạp từ điển từ file (có thể nạp tự động khi chạy chương trình)

b. Tra từ điển (nhập vào một từ, hiển thị nghĩa của nó nếu có trong từ điển)

c. Sửa đổi từ

d. Thêm từ mới

e. Xóa từ

f. Lưu từ điển vào fil

**1.2 Phân tích bài toán**

**-Xác định các yêu cầu của bài toán, xác định các lớp, các thuộc tính, các phương thức của lớp:**

+Yêu cầu của bài toán (các chức năng):

1. Nạp danh sách từ một tệp văn bản (fileToHashTable).
2. Tìm kiếm nghĩa của một từ trong danh sách (findOnHashTable).
3. Sửa đổi nghĩa của một từ (fixWord).
4. Thêm một từ mới vào danh sách (addWord).
5. Xóa một từ khỏi danh sách (removeWord).
6. Lưu danh sách từ điển vào một tệp văn bản (saveDicToFile).

+Phân loại thành các lớp, thuộc tính và phương thức:

1. **Lớp Hashtable:**

+**Thuộc tính**:

head: Trỏ đến nút đầu tiên trong danh sách liên kết.

+ **Phương thức**:

**fileToHashTable**: Nạp danh sách từ tệp văn bản

**findOnHashTable:** Tìm kiếm nghĩa của một từ trong danh sách.

**fixWord**: Sửa đổi nghĩa của một từ.

**addWord**: Thêm một từ mới vào danh sách.

**removeWord:** Xóa một từ khỏi danh sách.

**saveDicToFile**: Lưu danh sách từ điển vào tệp văn bản.

Các phương thức hỗ trợ khác cho quá trình quản lý danh sách.

1. **Lớp Node (nội tại trong Hashtable):**

+**Thuộc tính**:

**key**: Lưu trữ từ tiếng Anh

**value**: Lưu trữ nghĩa của từ

**next**: Trỏ đến nút tiếp theo trong danh sách liên kết.

1. **Hàm main:**

Thực hiện việc sử dụng lớp **Hashtabl**e để quản lý từ điển trong chương trình.

-Mô tả chức năng của từng lớp, từng phương thức:

**Lớp Hashtable:**

**Thuộc tính:**

+**head**: Là con trỏ đầu tiên của danh sách liên kết, trỏ đến nút đầu của danh sách từ điển.

**Phương thức:**

**1. fileToHashTable(const string& filename)**:

+**Mô tả**: Phương thức này được sử dụng để nạp danh sách từ một tệp văn bản.

+**Thực hiện**: Đọc tệp văn bản với tên filename, chia các dòng thành cặp (key, value) dựa trên dấu hai chấm (":") và thêm chúng vào danh sách băm.

Điểm cần cải thiện: Cần kiểm tra nếu danh sách băm không rỗng thì cần giải phóng bộ nhớ trước khi nạp từ tệp mới.

**2. findOnHashTable(string word):**

Mô tả: Phương thức này tìm kiếm nghĩa của một từ trong danh sách từ điển.

Thực hiện: Duyệt qua danh sách từ đầu đến cuối, so sánh từ khóa (key) với từ cần tìm. Nếu tìm thấy, in ra nghĩa của từ đó.

Điểm cần cải thiện: Không cần duyệt hết danh sách nếu đã tìm thấy từ cần tìm.

**3. fixWord(string wordNeedFix, string wordFix)**:

Mô tả: Phương thức này sửa đổi nghĩa của một từ trong danh sách từ điển.

Thực hiện: Tìm từ cần sửa, nếu tìm thấy, thay đổi nghĩa của từ thành nghĩa mới.

Điểm cần cải thiện: Cần cung cấp thông báo nếu từ không tồn tại trong từ điển.

**4. addWord(string word, string meanOfWord):**

Mô tả: Phương thức này thêm một từ mới vào danh sách từ điển.

Thực hiện: Tạo một nút mới và thêm vào cuối danh sách.

Điểm cần cải thiện: Cần xử lý trường hợp danh sách rỗng.

**5. removeWord(string word):**

Mô tả: Phương thức này xóa một từ khỏi danh sách từ điển.

Thực hiện: Tìm từ cần xóa, nếu tìm thấy, xóa nút chứa từ đó khỏi danh sách.

Điểm cần cải thiện: Cần cung cấp thông báo nếu từ không tồn tại trong từ điển.

**6. saveDicToFile(const string& filename):**

Mô tả: Phương thức này lưu danh sách từ điển vào một tệp văn bản.

Thực hiện: Ghi danh sách từ điển ra tệp văn bản có tên filename.

Điểm cần cải thiện: Cần kiểm tra trạng thái ghi tệp và cung cấp thông báo lỗi nếu không thể ghi.

**Lớp Node (nội tại trong Hashtable)**:

**Thuộc tính:**

key: Lưu trữ từ tiếng Anh.

value: Lưu trữ nghĩa của từ.

next: Trỏ đến nút tiếp theo trong danh sách liên kết.

**Hàm main**:

Mô tả: Hàm main được sử dụng để thực hiện việc sử dụng lớp Hashtable để quản lý từ điển trong chương trình.

**1.4 Phân tích thời gian chạy của từng phương thức có trong lớp**

**fileToHashTable:**

Thời gian chạy của phương thức này phụ thuộc vào số lượng dòng trong file và độ dài của mỗi dòng.

Độ phức tạp: O(n \* m), trong đó n là số dòng trong file, m là độ dài trung bình của mỗi dòng.

**findOnHashTable**:

Phương thức này truy cập danh sách liên kết từ đầu đến cuối để tìm một từ.

Độ phức tạp: O(n), trong đó n là số nút trong danh sách liên kết.

**fixWord:**

Phương thức này tìm một từ trong danh sách liên kết và cập nhật giá trị của nó.

Độ phức tạp: O(n), trong đó n là số nút trong danh sách liên kết.

**addWord**:

Phương thức này thêm một từ mới vào danh sách liên kết.

Độ phức tạp: O(1), vì bạn thêm vào cuối danh sách liên kết.

**removeWord:**

Phương thức này xóa một từ khỏi danh sách liên kết.

Độ phức tạp: O(n), trong trường hợp xóa từ ở giữa danh sách.

**saveDicToFile:**

Phương thức này duyệt danh sách liên kết và ghi dữ liệu vào file.

Độ phức tạp: O(n), trong đó n là số nút trong danh sách liên kết.

Lưu ý rằng đây chỉ là ước tính về độ phức tạp thời gian tùy thuộc vào cách triển khai cụ thể và dữ liệu đầu vào. Để có phân tích cụ thể hơn, bạn cần sử dụng độ phức tạp Big O notation cho từng phương thức dựa trên cách bạn triển khai chúng và dữ liệu đầu vào cụ thể

**2.1 Đề bài**

1. Cài đặt cấu trúc dữ liệu trừu tượng danh sách liên kết kép và các bộ duyệt xuôi và ngược.

2. Lập chương trình quản lý danh sách học sinh, chương trình có những chức năng sau: (Hồ sơ một

học sinh giả sử có: Tên, lớp, số điện thoại, điểm TB)

a. Nhập danh sách học sinh từ bàn phím hay từ file.

b. In ra danh sách học sinh gồm có tên và xếp loại

c.In ra danh sách học sinh gồm các thông tin đầy đủ

d. Nhập vào từ bàn phím một tên học sinh và một tên lớp, tìm xem có học sinh có tên nhập vào

trong lớp đó không? Nếu có thì in ra số điện thoại của học sinh đó

e. Nhập vào một hồ sơ học sinh mới từ bàn phím, bổ sung học sinh đó vào danh sách.

f. Nhập vào từ bàn phím tên một lớp, loại bỏ tắt cả các học sinh của lớp đó khỏi danh sách.

g. Sắp xếp danh sách học sinh theo thứ tự giảm dần của điểm trung bình.

h. Nhập vào hồ sơ một học sinh mới từ bàn phím, chèn học sinh đó vào danh sách mà không làm

thay đổi thứ tự đã sắp xếp, in ra danh sách mới.

i. Lưu trữ lại trên đĩa danh sách học sinh khi đã thay đổi.

**2.1.2 Phân tích bài toán**

**Xác định các yêu cầu của bài toán, xác định các lớp, các thuộc tính, các phương thức của lớp.**

**Mô tả chức năng của từng lớp, từng phương thức**

1. **Lớp Node:**

+ **Mô tả**: Lớp Node đại diện cho các nút trong danh sách liên kết đôi. Mỗi nút chứa dữ liệu (kiểu T), một con trỏ tới nút trước (prev) và một con trỏ tới nút tiếp theo (next).

+**Các phương thức:**

**Node()**: Hàm tạo mặc định không chứa dữ liệu.

**Node(T data)**: Hàm tạo với dữ liệu được truyền vào.

Các getter và setter để truy cập và cập nhật dữ liệu, nút trước và nút tiếp theo.

**2.Lớp DbLinkedList:**

+**Mô tả**: Lớp này đại diện cho danh sách liên kết đôi và bao gồm các nút Node.

**+Các thuộc tính:**

head: Con trỏ đến nút đầu danh sách.

tail: Con trỏ đến nút cuối danh sách.

n: Số lượng nút trong danh sách.

**+Các phương thức**:

**DbLinkedList()**: Hàm tạo một danh sách liên kết đôi rỗng.

**insertInLast(T data)**: Thêm một nút mới chứa dữ liệu data vào cuối danh sách.

**insertInFirst(T data):** Thêm một nút mới chứa dữ liệu data vào đầu danh sách.

**eraseFirstNode()**: Xóa nút đầu tiên trong danh sách.

**eraseLastNode()**: Xóa nút cuối cùng trong danh sách.

**3.Lớp Iterator:**

+**Mô tả**: Lớp này được nhúng trong lớp DbLinkedList và đại diện cho một vị trí trong danh sách liên kết đôi.

**+Các thuộc tính**:

current: Con trỏ tới nút hiện tại mà Iterator đang trỏ đến.

**+Các phương thức**:

Các toán tử (\*, ++, --, ==, !=) để hỗ trợ duyệt danh sách và so sánh Iterator.

Trong hàm main(), bạn tạo một đối tượng DbLinkedList và thực hiện các thao tác trên danh sách, sau đó duyệt danh sách xuôi và ngược bằng cách sử dụng Iterator.

Chức năng của mã là xây dựng danh sách liên kết đôi và thực hiện các thao tác cơ bản như thêm và xóa nút, sau đó duyệt danh sách xuôi và ngược.

**2.1.4 Phân tích thời gian chạy của từng phương thức có trong các lớp**

* **insertInLast và insertInFirst trong lớp DbLinkedList**:

Độ phức tạp: O(1)

Cả hai phương thức này thêm một nút vào danh sách liên kết ở vị trí cố định (đầu hoặc cuối danh sách), do đó thời gian thêm nút không phụ thuộc vào kích thước danh sách. Do đó, độ phức tạp là O(1).

* **eraseFirstNode và eraseLastNode trong lớp DbLinkedList:**

Độ phức tạp: O(1)

Cả hai phương thức này xóa một nút khỏi danh sách liên kết ở vị trí cố định (đầu hoặc cuối danh sách), do đó thời gian xóa nút không phụ thuộc vào kích thước danh sách. Do đó, độ phức tạp là O(1).

* **begin và end trong lớp DbLinkedList:**

Độ phức tạp: O(1)

Cả hai phương thức này chỉ trả về Iterator cho đầu và cuối danh sách, và việc truy cập các con trỏ này có độ phức tạp O(1).

* **rbegin và rend trong lớp DbLinkedList:**

Độ phức tạp: O(1)

Tương tự, cả hai phương thức này chỉ trả về Iterator cho đầu và cuối danh sách ngược (reverse), và việc truy cập các con trỏ này có độ phức tạp O(1).

* **Duyệt danh sách bằng Iterator:**

Độ phức tạp: O(n)

Duyệt danh sách bằng Iterator có độ phức tạp tuyến tính với số lượng nút trong danh sách (n). Do đó, thời gian duyệt danh sách là O(n).

**2.2.2 Phân tích bài toán**

**Xác định các yêu cầu của bài toán, xác định các lớp, các thuộc tính, các phương thức của lớp. Mô tả chức năng của từng lớp, từng phương thức**

* **Yêu cầu của bài toán:**

1.Đọc danh sách sinh viên từ một tệp tin (file) có định dạng nhất định.

2.Cho phép nhập thông tin về sinh viên từ bàn phím.

3.In ra tên của sinh viên và xếp loại dựa trên điểm trung bình.

4.In ra toàn bộ thông tin của danh sách sinh viên.

5.Tìm kiếm thông tin của một sinh viên dựa trên tên và lớp.

6.Thêm một sinh viên mới vào danh sách.

7.Xóa tất cả các sinh viên thuộc một lớp khỏi danh sách.

8.Sắp xếp danh sách sinh viên theo điểm trung bình giảm dần.

9.Thêm một sinh viên mới và đảm bảo rằng danh sách vẫn được sắp xếp 10.theo điểm trung bình giảm dần.

11.Lưu danh sách sinh viên vào một tệp tin sau khi thay đổi.

* **Phân tích lớp Student:**

1. Lớp Student đại diện cho thông tin của một sinh viên. Các thành phần của lớp này bao gồm:

* **Các thuộc tính:**

ten (tên của sinh viên)

lop (lớp của sinh viên)

sdt (số điện thoại của sinh viên)

dtb (điểm trung bình của sinh viên)

* **Các phương thức**:

**getTen()**: Trả về tên của sinh viên.

**getLop():** Trả về lớp của sinh viên.

**getSdt():** Trả về số điện thoại của sinh viên.

**getDtb()**: Trả về điểm trung bình của sinh viên.

**setTen()**: Đặt tên của sinh viên, kiểm tra nếu tên không được để trống.

**setLop()**: Đặt lớp của sinh viên, kiểm tra nếu lớp không được để trống.

**setSdt()**: Đặt số điện thoại của sinh viên.

**setDtb()**: Đặt điểm trung bình của sinh viên, kiểm tra nếu điểm nằm trong khoảng hợp lệ.

* **Phân tích lớp studentList:**

Lớp studentList đại diện cho danh sách sinh viên và thực hiện các hoạt động quản lý danh sách. Các thành phần của lớp này bao gồm:

* **Các thuộc tính:**

**Node**: Một lớp con (nested class) đại diện cho một nút trong danh sách liên kết, chứa dữ liệu Student.

**head**: Con trỏ đến đầu danh sách liên kết.

* **Các phương thức:**

**FileToList(const string& filename)**: Đọc danh sách sinh viên từ một tệp tin.

**inputStudentList()**: Cho phép người dùng nhập thông tin sinh viên từ bàn phím và thêm vào danh sách.

**printNameandRank()**: In ra tên của sinh viên và xếp loại dựa trên điểm trung bình.

**printAll():** In ra toàn bộ thông tin của danh sách sinh viên.

**findStudent(const string& student, const string& classroom)**: Tìm một sinh viên dựa trên tên và lớp.

**addStudent()**: Thêm một sinh viên mới vào danh sách.

**delStudent(const string& classroom)**: Xóa tất cả sinh viên thuộc một lớp khỏi danh sách.

**sortDecending()**: Sắp xếp danh sách theo điểm trung bình giảm dần.

**addAndSort()**: Thêm một sinh viên mới và đảm bảo danh sách vẫn được sắp xếp theo điểm trung bình giảm dần.

**saveToFile(const string& filename):** Lưu danh sách sinh viên vào tệp tin sau khi thay đổi.

**2.2.4:Phân tích thời gian chạy của từng phương thức có trong các lớp**

* **Trong lớp Student:**

**setTen(const string& ten)**: Phương thức này kiểm tra xem tên có bị để trống không. Điều này yêu cầu kiểm tra mỗi ký tự trong chuỗi tên. Do đó, thời gian chạy của phương thức này là O(n), trong đó n là độ dài của chuỗi tên.

**setLop(const string& lop)**: Phương thức này tương tự như setTen và cũng có thời gian chạy O(n), với n là độ dài của chuỗi lớp.

**setDtb(float dtb)**: Phương thức này kiểm tra điểm trung bình có nằm trong khoảng hợp lệ không, do đó thời gian chạy của phương thức này là O(1) vì không phải kiểm tra chuỗi.

* **Trong lớp studentList:**

**FileToList(const string& filename)**: Phương thức này đọc danh sách sinh viên từ một tệp tin. Thời gian chạy của phương thức này phụ thuộc vào số lượng sinh viên trong tệp tin và là O(m), với m là số lượng sinh viên trong tệp tin.

**inputStudentList():** Phương thức này cho phép người dùng nhập thông tin sinh viên từ bàn phím. Thời gian chạy phụ thuộc vào số lượng sinh viên mà người dùng muốn nhập, vì vậy thời gian chạy có thể là O(k), với k là số lượng sinh viên được nhập.

**printNameandRank()**: Phương thức này in ra tên của sinh viên và xếp loại dựa trên điểm trung bình. Thời gian chạy của phương thức này phụ thuộc vào số lượng sinh viên trong danh sách và là O(n), với n là số lượng sinh viên.

**printAll():** Phương thức này in ra toàn bộ thông tin của danh sách sinh viên. Thời gian chạy cũng phụ thuộc vào số lượng sinh viên và là O(n).

**findStudent(const string& student, const string& classroom)**: Phương thức này tìm một sinh viên dựa trên tên và lớp. Thời gian chạy phụ thuộc vào vị trí của sinh viên cần tìm và là O(n), với n là số lượng sinh viên.

**addStudent()**: Phương thức này thêm một sinh viên mới vào danh sách. Thời gian chạy của phương thức này cũng phụ thuộc vào vị trí thêm sinh viên và là O(1) hoặc O(n), tùy thuộc vào cách thêm vào cuối danh sách.

**delStudent(const string& classroom)**: Phương thức này xóa tất cả sinh viên thuộc một lớp khỏi danh sách. Thời gian chạy phụ thuộc vào số lượng sinh viên thuộc lớp đó và là O(n).

**sortDecending()**: Phương thức này sắp xếp danh sách theo điểm trung bình giảm dần. Thời gian chạy của phương thức sắp xếp sử dụng thuật toán Bubble Sort, và trong trường hợp xấu nhất là O(n^2) (tương đương với sắp xếp nhanh).

**addAndSort()**: Phương thức này thêm một sinh viên mới và đảm bảo danh sách vẫn được sắp xếp theo điểm trung bình giảm dần. Thời gian chạy của phương thức này phụ thuộc vào vị trí thêm sinh viên và sắp xếp, và là O(n) hoặc O(1).

**saveToFile(const string& filename)**: Phương thức này lưu danh sách sinh viên vào một tệp tin sau khi thay đổi. Thời gian chạy phụ thuộc vào số lượng sinh viên và là O(n).