TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

----------o0o----------

Bài tập lớn môn học

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Giảng viên hướng dẫn: TS. Hoàng Văn Thông

Sinh viên thực hiện: Ngô Thị Thanh – 231230902

Lớp CNTT2-K64

Đề bài số: 19

Hà Nội tháng 11 năm 2024

Mục lục

LỜI MỞ ĐẦU

Trong thời đại công nghệ phát triển, việc quản lý thông tin sinh viên một cách hiệu quả là yêu cầu quan trọng đối với các trường học và tổ chức giáo dục. Đề tài "Quản lý danh sách sinh viên" được thực hiện với mục tiêu xây dựng một hệ thống lưu trữ thông tin sinh viên tiện dụng, giúp người dùng dễ dàng thực hiện các thao tác thêm, xóa, tìm kiếm và sắp xếp danh sách sinh viên theo các tiêu chí như tên, lớp và điểm trung bình.

Bên cạnh đó, để tối ưu hóa quá trình tìm kiếm, chương trình áp dụng cấu trúc dữ liệu Trie – một loại cây đặc biệt được thiết kế để lưu trữ và tra cứu các từ có chung tiền tố nhanh chóng. Việc kết hợp Trie trong quản lý danh sách sinh viên giúp cải thiện hiệu suất tìm kiếm, đặc biệt khi cần xác định số lượng từ có chung tiền tố.

Báo cáo này sẽ trình bày quá trình xây dựng chương trình từ phân tích, thiết kế đến cài đặt, cùng với những đánh giá về tính hiệu quả và ứng dụng thực tiễn của chương trình. Qua đó, chúng tôi mong muốn mang đến một giải pháp quản lý thông tin sinh viên thuận tiện, chính xác và tối ưu cho người dùng.

Phần A. BÀI TOÁN QUẢN LÝ

I. Đề bài

Ứng dụng lớp vector để quản lý các đối tượng là các sinh viên. Mỗi sinh viên cần quản lý các thông tin sau: Mã SV, Họ tên, Ngày sinh, giới tính, lớp, GPA. Trong lớp sinh viên xây dựng toán tử so sánh theo họ tên.

Viết chương trình cho phép thực hiện các chức năng sau:

- Nhập thêm các danh sách sinh viên, sau khi hoàn thành nhập thông tin của một sinh viên, chương trình đưa ra câu hỏi có nhập nữa không (c/k)? Nếu người dùng nhập: c thì tiếp tục nhập, nhập k thì kết thúc.
- Xóa đi một sinh viên o Sửa đổi thông tin của một sinh viên bất kỳ trong danh sách.
- Sắp xếp danh sách sinh viên theo họ tên.
- Hiển thị toàn bộ danh sách sinh viên hiện có trong vector.
- Tìm kiếm sinh viên theo họ tên.

II. Phân tích bài toán

1. Khái niệm Vector

Khái niệm Vector: Kiểu dữ liệu trừu tượng Vector là sự mở rộng của khái niệm mảng. Vector là một mảng lưu trữ một dãy các đối tượng với số lượng tùy ý.

2. Yêu cầu bài toán

Chương trình cần đáp ứng các yêu cầu sau:

- Nhập thông tin sinh viên: Nhập các thông tin như họ tên, ngày sinh, giới tính, lớp học và điểm GPA.
- Hiển thị danh sách sinh viên: Xuất danh sách sinh viên theo các thông tin đã nhập.
- Xóa sinh viên: Cung cấp chức năng xóa sinh viên theo mã số sinh viên.
- Sửa thông tin sinh viên: Cập nhật lại thông tin sinh viên theo mã số sinh viên.
- Sắp xếp danh sách sinh viên: Sắp xếp danh sách sinh viên theo tên.
- Tìm kiếm sinh viên theo tên: Tìm kiếm sinh viên theo tên.

3. Các lớp và thuộc tính

Lớp SV:

- **Thuộc tính**:
  - *maSV*: Mã số sinh viên.
  - *hoTen*: Họ và tên sinh viên.
  - *ngaySinh, thangSinh, namSinh*: Ngày tháng năm sinh.
  - *gioiTinh*: Giới tính sinh viên.
  - *lop*: Lớp học của sinh viên.
  - *GPA*: Điểm GPA của sinh viên.
- **Phương thức**:
  - ***operator*>>**: Nhập thông tin sinh viên từ bàn phím.
  - ***operator*<<**: Xuất thông tin sinh viên ra màn hình.
  - ***operator*<**: So sánh hai sinh viên dựa trên họ tên.
  - ***getMaSV(), getHoTen(), getGPA()***: Trả về các thông tin cụ thể của sinh viên.

Lớp DS_SV:

- **Thuộc tính**:
  - *SinhVien*: Mảng lưu trữ các đối tượng sinh viên.
- **Phương thức**:
  - ***nhapDanhSach()*:** Nhập danh sách sinh viên.
  - ***themSinhVien(SV& sv)*:** Thêm sinh viên vào danh sách.
  - ***xuatDS()*:** Xuất danh sách sinh viên.
  - ***sapxepDS()*:** Sắp xếp danh sách sinh viên theo tên.
  - ***timGPAMax(), timGPAMin()*:** Tìm sinh viên có GPA cao nhất và thấp nhất.
  - ***xoaSV(int& id)*:** Xóa sinh viên theo mã số sinh viên.
  - ***suaThongTinSV(int& id)*:** Sửa thông tin sinh viên theo mã số sinh viên.

- ***timKiemSV(string& name):*** Tìm kiếm sinh viên theo tên.
- ***ktraDSRong():*** Kiểm tra danh sách có rỗng không.

Lớp APP:

- **Thuộc tính**:
  - *qlySinhVien*: Đối tượng quản lý danh sách sinh viên.
- **Phương thức**:
  - ***menu():*** Hiển thị menu và thực hiện các thao tác tương ứng với lựa chọn của người dùng.

III. Cài đặt các lớp và hàm main

Chương trình sử dụng C++ với ba lớp chính: SV, DS_SV, và APP.

Tham khảo code đầy đủ tại: https://github.com/ngothithanhh/CTDLvaGT/blob/main/BAITAPLON/Bai_toan_quan_ly/QL_SinhVien_Vector.cpp

Lớp SV:

Dùng để đại diện cho một sinh viên, chứa các thuộc tính và phương thức cần thiết để thao tác với sinh viên.

```
class SV{
  private:
    int maSV;
    string hoTen;
    int ngaySinh;
    int thangSinh;
    int namSinh;
    string gioiTinh;
    string lop;
    float GPA;

  public:
    SV(){}
    friend istream& operator>>(istream& is, SV& sv){
      cout << "Nhap thong tin sinh vien:\n";
      cout << "\tNhap ho va ten: "; is.ignore(); getline(is, sv.hoTen);
      cout << "\tNhap ngay thang nam sinh: "; is >> sv.ngaySinh >> sv.thangSinh >> sv.namSinh;
      cout << "\tNhap gioi tinh: "; is.ignore(); getline(is, sv.gioiTinh);
      cout << "\tNhap lop: "; is >> sv.lop;
      cout << "\tNhap diem tich luy: "; is >>sv.GPA;
      return is;

    }

    friend ostream& operator<<(ostream& os, SV& sv){
      os << "Thong tin sinh vien: \n";
      os << "\tHo va ten: " << sv.hoTen << "\n";
      os << "\tNgay sinh: " << sv.ngaySinh <<" / " << sv.thangSinh << " / " << sv.namSinh << "\n";
      os << "\tGioi tinh: " << sv.gioiTinh << "\n";
      os << "\tLop: " << sv.lop << "\n";
      os << "\tDiem tich luy: " << sv.GPA << "\n";
      return os;
    }

    bool operator<(const SV& other) const {
    return hoTen < other.hoTen;
    }

    int getMaSV(){
      return maSV;
    }
    string getHoTen(){
      return hoTen;
    }
    float getGPA(){
      return GPA;
```

```
    }
};
```

Lớp DS_SV:

Quản lý danh sách sinh viên, với các phương thức để thêm, xóa, sửa, tìm kiếm, và sắp xếp sinh viên, kiểm tra danh sách rỗng.

```
class DS_SV{
  private:
    vector<SV> SinhVien;

  public:

    //Nhap danh sach sinh vien
    void nhapDanhSach() {
      char tiepTuc = 'c';
      while (tiepTuc == 'c') {
        SV sv;
        cin >> sv;
        SinhVien.push_back(sv);
        cout << "Co tiep tuc nhap nua khong? (c/k): ";
        cin >> tiepTuc;
      }
    }

    //Them sinh vien vao danh sach
    void themSinhVien(SV& sv){
      SinhVien.push_back(sv);
    }

    //Hien thi danh sach sinh vien
    void xuatDS(){
      for(auto& sv: SinhVien){
        cout << sv << "\n";
      }
    }

    //Sap xep danh sach sinh vien theo ho va ten
    void sapxepDS(){
      sort(SinhVien.begin(), SinhVien.end());
    }

    //Tim GPA cao nhat
    SV timGPAMax(){
      return *max_element(SinhVien.begin(), SinhVien.end());
    }

    //Tim GPA thap nhat
    SV timGPAMin(){
      return *min_element(SinhVien.begin(), SinhVien.end());
    }
```

```
    //Xoa sinh vien theo ma sinh vien
    void xoaSV(int& id){
       for(auto it = SinhVien.begin(); it != SinhVien.end(); ++it){
          if(it->getMaSV() == id){
             SinhVien.erase(it);
             cout << "Da xoa !!!\n";
             return;
          }
       }

       cout << "Khong tim thay !!\n";
    }

    //Sua thong tin sinh vien theo ma sinh vien
    void suaThongTinSV(int& id){
       for(auto& sv: SinhVien){
          if(sv.getMaSV() == id){
             cout << "Nhap thong tin moi cho sinh vien:\n";
             cin >> sv;
             return;
          }
       }
       cout << "Khong tim thay sinh vien nay!!\n";
    }

    //Tim kiem thong tin sinh vien theo ho ten
    void timKiemSV(string& name){
       int check = 0;
       for(auto& sv : SinhVien){
          if(sv.getHoTen() == name){
             cout << "Thong tin sinh vien co ten " << name <<" la:\n";
             cout << sv;
             check++;
          }
       }
       if(!check) cout << "Khong tim thay sinh vien co ten: " << name << "\n";
    }

    //Kiem tra danh sach rong?
    bool ktraDSRong(){
       return SinhVien.empty();
    }

};
```

Lớp APP:

Quản lý giao diện người dùng, cung cấp menu để người dùng lựa chọn thao tác với danh sách sinh viên.

```
class APP{
  private:
    DS_SV qlySinhVien;

  public:
    void menu(){
      int choice;
      do {
        cout << "\n===== Menu Quan Ly Sinh Vien =====\n";
        cout << "1. Nhap danh sach sinh vien\n";
        cout << "2. Hien thi danh sach sinh vien\n";
        cout << "3. Xoa sinh vien\n";
        cout << "4. Sua thong tin sinh vien\n";
        cout << "5. Sap xep sinh vien theo ho ten\n";
        cout << "6. Tim kiem sinh vien theo ho ten\n";
        cout << "0. Thoat\n";
        cout << "Nhap lua chon: ";
        cin >> choice;

        switch (choice) {
          case 1:
            qlySinhVien.nhapDanhSach();
            break;

          case 2:{
            if(qlySinhVien.ktraDSRong()){
              cout << "Danh sach sinh vien hien dang rong!!\n";
            }
            else{
              qlySinhVien.xuatDS();
            break;

            }
          }

          case 3: {
            if(qlySinhVien.ktraDSRong()){
              cout << "Danh sach sinh vien hien dang rong!!\n";
            }
            else{
              int maSV;
              cout << "Nhap ma sinh vien can xoa: ";
              cin >> maSV;
              qlySinhVien.xoaSV(maSV);
```

```
                break;
            }
        }

        case 4: {
            if(qlySinhVien.ktraDSRong()){
                cout << "Danh sach sinh vien hien dang rong!!\n";
            }
            else{
                int maSV;
                cout << "Nhap ma sinh vien can sua: ";
                cin >> maSV;
                qlySinhVien.suaThongTinSV(maSV);
                break;
            }
        }

        case 5:
            qlySinhVien.sapxepDS();
            break;

        case 6: {
            if(qlySinhVien.ktraDSRong()){
                cout << "Danh sach sinh vien hien dang rong!!\n";
            }
            else{
                string hoTen;
                cout << "Nhap ten sinh vien can tim: ";
                cin.ignore();
                getline(cin, hoTen);
                qlySinhVien.timKiemSV(hoTen);
                break;
            }
        }

        case 0:
            cout << "Thoat chuong trinh!!!\n";
            break;

        default:
            cout << "Lua chon khong hop le!!\n";
            break;
      }
    } while (choice != 0);
  }
};
```

Hàm main:

```
main(){
  APP app;
  app.menu();
  return 0;
```

```
}
```

IV. Phân tích thời gian chạy của chương trình

1. nhapDanhSach():

- Mỗi lần nhập một sinh viên là thao tác O(1), nhưng phải nhập nhiều sinh viên trong khi người dùng chọn tiếp tục nhập, do đó, thời gian tổng thể phụ thuộc vào số lượng sinh viên được nhập vào.

2. themSinhVien(SV& sv):

- Thêm một sinh viên vào danh sách mất O(1) thời gian.

3. xuatDS():

- Duyệt qua tất cả các sinh viên trong danh sách và in thông tin của họ, có độ phức tạp O(n), trong đó n là số lượng sinh viên trong danh sách.

4. sapxepDS():

- Sắp xếp danh sách sinh viên theo họ tên, sử dụng thuật toán sắp xếp nổi bật là QuickSort hoặc MergeSort với độ phức tạp O(n log n).

5. timKiemSV(string& name):

- Tìm kiếm sinh viên theo tên yêu cầu duyệt qua tất cả các sinh viên trong danh sách, độ phức tạp O(n).

6. xoaSV(int& id):

- Xóa sinh viên theo mã số yêu cầu tìm kiếm sinh viên trong danh sách và sau đó xóa, có độ phức tạp O(n).

7. suaThongTinSV(int& id):

- Cập nhật thông tin sinh viên theo mã số yêu cầu tìm kiếm và sửa, có độ phức tạp O(n).

V. Kết luận

Chương trình quản lý sinh viên được cài đặt hoàn chỉnh với đầy đủ các chức năng cần thiết như nhập, xuất, tìm kiếm, sửa, xóa và sắp xếp danh sách sinh viên. Việc sử dụng các lớp đối tượng để quản lý sinh viên và danh sách sinh viên giúp chương trình dễ dàng mở rộng và duy trì.

Các thao tác chính trong chương trình có độ phức tạp hợp lý, phù hợp với yêu cầu bài toán về quản lý danh bạ sinh viên. Trong thực tế, với lượng sinh viên lớn, chương trình vẫn hoạt động hiệu quả nhờ vào các thuật toán sắp xếp và tìm kiếm tối ưu.

VI. Danh sách tài liệu tham khảo

- GeeksforGeeks – C++ STL documentation

Phần B. Bài 19

I. Đề bài

http://laptrinhonline.club/problem/tichpxdanhba

Tạo một cây từ điển để giải bài toán xây dựng danh bạ sau:

Tèo muốn xây dựng danh bạ liên lạc để lưu vào trong máy tính. Trong quá trình xây dựng danh bạ Tèo để ý và thấy rằng rất nhiều từ nọ là tiền tố của từ kia. Nếu từ x được ghép bởi hai từ y và z tức là x=yz thì y được gọi là tiền tố và z gọi là hậu tố của x (Ví dụ từ tichpx có các tiền tố t, ti, tic, tich, tichp, tichpx). Trong khi cập nhật danh bạ với mỗi từ Tèo có 1 trong hai thao tác:

- add là bổ sung một liên lạc vào danh bạ.
- find là tìm xem một từ là tiền tố của bao nhiêu liên lạc đã có Bạn hãy lập trình để xác định giúp Tèo nhé.

Input:

- Dòng đầu chứa số nguyên dương n là số thao tác (1<=n<=105) (1<=n<=105).
- n dòng tiếp theo mỗi dòng thuộc một trong hai thao tác hoặc là add hoặc là find một liên lạc là một xâu gồm toàn chữ thường tiếng anh có độ dài không quá 30 ký tự.

Output: Với mỗi thao tác find xuất ra màn hình số liên lạc đã có trong danh bạ có tiền tố muốn tìm

Ví dụ:

Input:	Output
6 add daihocgiaothongvantai add daihocthuyloi find daihoc add daihocxaydung find hocvien find dai	2 0 3

II. Phân tích bài toán

1. Khái niệm Trie

Trie data cấu trúc là một cây dạng dữ liệu cấu trúc được sử dụng để lưu trữ một tập hợp chuỗi động. Nó thường được sử dụng để truy xuất và lưu trữ các khóa hiệu quả trong một tệp lớn. Cấu trúc này hỗ trợ các hoạt động như chèn , tìm kiếm và xóa khóa, khiến nó trở thành một công cụ có giá trị trong các lĩnh vực như khoa học máy tính và truy xuất thông tin. Trong bài viết này, chúng tôi sẽ khám phá hoạt động chèn và tìm kiếm trong Trie cấu trúc dữ liệu.

2. Yêu cầu bài toán

Cần cài đặt một hệ thống danh bạ, hỗ trợ hai thao tác:

- **add**: Thêm một từ vào danh bạ.
- **find**: Tìm kiếm số lượng từ có tiền tố là một chuỗi cho trước.

Mỗi từ chỉ chứa các chữ cái thường từ 'a' đến 'z' và độ dài không quá 30 ký tự.

Cần tối ưu hóa thời gian thực hiện thao tác tìm kiếm tiền tố, bởi vì số lượng thao tác có thể lên đến 100,000.

3. Các lớp, thuộc tính và phương thức

Lớp Node:

- **Thuộc tính**:
  - *children[26]:* Mảng các con trỏ con, mỗi con trỏ tương ứng với một ký tự trong bảng chữ cái.
  - *count:* Đếm số lần một từ đi qua node này (hoặc số lần tiền tố xuất hiện).
- **Phương thức**:
  - *Constructor Node():* Khởi tạo các con trỏ con và count bằng 0.

Lớp Trie:

- **Thuộc tính**:
  - *root:* Con trỏ tới node gốc của Trie.
- **Phương thức**:
  - *Constructor Trie():* Khởi tạo Trie với node gốc.

- *add(const string& str):* Thêm một từ vào Trie. Duyệt qua từng ký tự của từ và tạo các node con nếu cần.
- *find(const string& fin):* Tìm số lần tiền tố xuất hiện trong Trie. Duyệt qua từng ký tự của tiền tố và trả về giá trị *count* của node cuối cùng.

4. Mô tả chức năng của từng lớp và phương thức

Lớp Node:

- Là lớp đại diện cho một node trong Trie, nơi lưu trữ thông tin của các ký tự trong từ.
- Mỗi node có thể chứa một hoặc nhiều node con, mỗi node con tương ứng với một ký tự trong từ.
- *count* là một thuộc tính dùng để đếm số lần mà tiền tố đi qua node này, giúp việc tìm kiếm tiền tố trở nên dễ dàng hơn.

Lớp Trie:

- Quản lý toàn bộ cấu trúc cây Trie, hỗ trợ các thao tác thêm từ và tìm kiếm tiền tố.
- Phương thức *add* giúp thêm từ vào Trie. Mỗi ký tự của từ sẽ tạo ra một node con nếu chưa có. Sau khi thêm xong, các node sẽ cập nhật giá trị count để theo dõi số lần từ đi qua chúng.
- Phương thức *find* giúp tìm kiếm số lần xuất hiện của một tiền tố. Duyệt qua từng ký tự của tiền tố và trả về giá trị count của node tương ứng với ký tự cuối cùng trong tiền tố.

III. Cài đặt các lớp và hàm main bằng C++

Code đầy đủ tại:
https://github.com/ngothithanhh/CTDLvaGT/blob/main/BAITAPLON/Bai_toan_xay_dung_danh_ba/Trie.cpp

Lớp Node:

```
class Node{
   public:
      Node* child[26]; //Moi phan tu la 1 not con cho cac ki tu tu 'a' den 'z'
      int count;

      Node(){
         count = 0;
         for(int i = 0; i < 26; i++){
            child[i] = nullptr;
         }
      }

};
```

Lớp Trie:

```
class Trie{
   private:
      Node *root;
   public:
      Trie(){
         root = new Node();
      }

      void add(const string& str) {
         Node* node = root;
         for (char c : str) {
            int index = c - 'a'; // Tinh chi so cua ky tu tu 0 den 25
            if (node->child[index] == nullptr) {
               node->child[index] = new Node();
            }
            node = node->child[index];
            node->count++;
         }
   }

      int find(const string& fin) {
         Node* node = root;
         for (char c : fin) {
            int index = c - 'a';
            if (node->child[index] == nullptr) {
               return 0; // Khong tim thay tien to
            }
            node = node->child [index];
         }
         return node->count; // Tra ve so tu co tien to can tim
      }
```

```
};
```

Hàm main:

```
main() {
   int n;
   cin >> n; //Nhap so cau lenh

   Trie trie;
   for (int i = 0; i < n; ++i) {
      string command, str;
      cin >> command >> str; // cu pha: lenh chuoikitu
      if (command == "add") {
         trie.add(str);
      } else if (command == "find") {
         cout << trie.find(str) << endl;
      }
   }
}
```

IV. Phân tích thời gian chạy của từng phương thức

1. add(const string& str):

Phân tích: Mỗi thao tác thêm một từ vào Trie cần duyệt qua từng ký tự của từ. Với mỗi ký tự, thời gian thực hiện là O(1) vì chỉ cần tính chỉ số của ký tự và kiểm tra hoặc tạo node con tương ứng.

Thời gian thực hiện: O(m), trong đó m là độ dài của từ.

2. find(const string& fin):

Phân tích: Mỗi thao tác tìm kiếm tiền tố trong Trie cần duyệt qua từng ký tự của tiền tố. Với mỗi ký tự, thời gian thực hiện là O(1) vì chỉ cần tính chỉ số của ký tự và di chuyển tới node con tương ứng.

Thời gian thực hiện: O(k), trong đó k là độ dài của tiền tố.

Tổng thời gian chạy cho n thao tác là O(n * m), trong đó m là độ dài trung bình của các từ trong danh bạ.

V. Danh sách tài liệu tham khảo

Giới thiệu về cấu trúc dữ liệu Trie trong C++: https://www.geeksforgeeks.org/trie-insert-and-search/

HowKteam: https://howkteam.vn/course/cau-truc-du-lieu-va-giai-thuat/trie-4329

KẾT LUẬN

Chương trình quản lý danh sách sinh viên kết hợp với cấu trúc dữ liệu Trie đã được xây dựng thành công, đáp ứng tốt các yêu cầu về quản lý, sắp xếp và tìm kiếm thông tin sinh viên. Nhờ áp dụng phương pháp lập trình hướng đối tượng và tối ưu hóa bằng cấu trúc Trie, chương trình có thể thực hiện các thao tác một cách nhanh chóng và chính xác, ngay cả khi danh sách sinh viên lớn.

Qua quá trình thử nghiệm, chương trình không chỉ đảm bảo tính chính xác và hiệu quả trong quản lý thông tin mà còn dễ sử dụng và có tính linh hoạt cao. Cấu trúc Trie, với khả năng hỗ trợ tìm kiếm nhanh chóng dựa trên tiền tố, giúp chương trình trở nên vượt trội trong việc xử lý các tác vụ tìm kiếm liên quan đến họ tên sinh viên hoặc từ khóa khác.

Với nền tảng hiện có, chương trình hoàn toàn có thể mở rộng thêm các tính năng mới như quản lý điểm số chi tiết, phân tích thống kê, hoặc tích hợp vào các hệ thống quản lý giáo dục. Đây là một bước tiến quan trọng trong việc ứng dụng công nghệ vào quản lý thông tin sinh viên một cách toàn diện và hiệu quả.

LỜI CẢM ƠN

Em xin chân thành cảm ơn giảng viên TS. Hoàng Văn Thông đã tận tình hướng dẫn và cung cấp những kiến thức quý báu trong môn học Cấu trúc dữ liệu và Giải thuật, là nền tảng quan trọng để hoàn thành đề tài này.