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: ThS. Phạm Xuân Tích

Sinh viên thực hiện: Nguyễn Việt Hoàng – Lớp CNTT3-K64

MSV: 223630669

Đề tài số: A B: 34

**Hà Nội, tháng 11 năm 2024**

**MỤC LỤC**

[**BÀI A:**  2](#_gjdgxs)

[**I.**](#_30j0zll) **Đề bài** 2

[**II.**](#_1fob9te) **Phân tích bài toán** 2

[**III.**](#_3znysh7) **Cài đặt các lớp và hàm main bằng C++** 9

[**IV.**](#_2et92p0) **Phân tích thời gian chạy của từng phương thức** 16

[**BÀI B:** 18](#_tyjcwt)

[**I.**](#_3dy6vkm) **Đề bài** 18

[**II.**](#_1t3h5sf) **Phân tích bài toán** 18

[**III.**](#_4d34og8) **Cài đặt các lớp và hàm main bằng C++** 22

[**IV.**](#_2s8eyo1) **Phân tích thời gian chạy của từng phương thức** 27

[**DANH SÁCH TÀI LIỆU THAM KHẢO** 3](#_17dp8vu)0

[**LỜI CẢM ƠN** 3](#_3rdcrjn)1

# **BÀI A**

1. **Đề bài**

Ứng dụng list để quản lý các đối tượng là sinh viên trong một trường đại học. Mỗi sinh viên cần quản lý các thông tin sau: Mã SV, Họ tên, Ngày sinh, Giới tính, Lớp học, và Điểm tổng kết. Trong lớp SinhVien, xây dựng toán tử so sánh để sắp xếp sinh viên theo điểm tổng kết và theo họ tên.

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

* Nhập danh sách sinh viên từ file: Đọc thông tin sinh viên từ file (theo định dạng quy định) và thêm vào danh sách hiện có.
* Ghi danh sách sinh viên vào file: Ghi danh sách sinh viên hiện tại vào file để lưu trữ.
* Nhập thêm sinh viên mới: Cho phép nhập một sinh viên mới từ bàn phím bằng cách sử dụng toán tử >>. Sau khi hoàn thành nhập thông tin của một sinh viên, chương trình hỏi người dùng có muốn nhập thêm không (c/k)? Nếu người dùng nhập c thì tiếp tục, nhập k thì dừng.
* Xóa một sinh viên: Tìm và xóa sinh viên khỏi danh sách theo mã sinh viên.
* Sửa đổi thông tin sinh viên: Cho phép người dùng nhập mã sinh viên cần sửa, sau đó nhập lại thông tin mới cho sinh viên đó.
* Sắp xếp danh sách sinh viên: Cho phép sắp xếp danh sách sinh viên theo điểm tổng kết hoặc họ tên (tuỳ lựa chọn của người dùng).
* Hiển thị danh sách sinh viên: Hiển thị toàn bộ danh sách sinh viên hiện có trong list.
* Tìm kiếm sinh viên theo tên: Tìm và hiển thị thông tin của sinh viên có tên trùng khớp với từ khoá tìm kiếm.
* Thống kê số lượng sinh viên: Hiển thị tổng số sinh viên hiện có trong danh sách.

1. **Phân tích bài toán**
2. Xác định các yêu cầu của bài toán

* Chương trình yêu cầu quản lý thông tin sinh viên với các tính năng như:
* Nhập thông tin sinh viên từ file và thêm mới sinh viên.
* Xóa, sửa và tìm kiếm sinh viên.
* Hiển thị danh sách sinh viên, sắp xếp danh sách và thống kê số lượng sinh viên.
* Lưu thông tin sinh viên vào file để dễ quản lý.

1. Xác định các lớp, các thuộc tính, các phương thức của lớp.
   1. LỚP “SinhVien”
      1. Mục đích

* Lớp đại diện cho một sinh viên với các thuộc tính cơ bản như mã sinh viên, tên, ngày sinh, giới tính, lớp và điểm tổng kết.
  + 1. Các thuộc tính
* maSV (string): Mã sinh viên.
* hoTen (string): Họ và tên của sinh viên.
* ngaySinh (string): Ngày sinh của sinh viên.
* gioiTinh (string): Giới tính của sinh viên.
* lopHoc (string): Lớp học của sinh viên.
* diemTongKet (double): Điểm tổng kết của sinh viên.
  + 1. Các phương thức
       1. Hàm tạo (constructor):
* Khởi tạo đối tượng SinhVien với đầy đủ thông tin.

*SinhVien*(string ms, string ht, string ns, string gt, string lh, double dtk) : *maSV*(ms), *hoTen*(ht), *ngaySinh*(ns), *gioiTinh*(gt), *lopHoc*(lh), *diemTongKet*(dtk) {}

* + - 1. Getter/Setter
* Các phương thức lấy và gán giá trị cho từng thuộc tính riêng biệt.

string *getMaSV*() *const* { *return* maSV; }

string *getHoTen*() *const* { *return* hoTen; }

string *getNgaySinh*() *const* { *return* ngaySinh; }

string *getGioiTinh*() *const* { *return* gioiTinh; }

string *getLopHoc*() *const* { *return* lopHoc; }

double *getDiemTongKet*() *const* { *return* diemTongKet; }

void *setHoTen*(string ht) { hoTen *=* ht; }

void *setNgaySinh*(string ns) { ngaySinh *=* ns; }

void *setGioiTinh*(string gt) { gioiTinh *=* gt; }

void *setLopHoc*(string lh) { lopHoc *=* lh; }

void *setDiemTongKet*(double dtk) { diemTongKet = dtk; }

* + - 1. Toán tử nhập/xuất (>>, <<)
* Nhập thông tin từ người dùng và hiển thị thông tin sinh viên.

*friend* istream*&* *operator>>*(istream*&* is, SinhVien*&* sv) {

cout *<<* "Nhap ma SV: "; is *>>* sv.maSV; is.*ignore*();

cout *<<* "Nhap ho va ten: "; *getline*(is, sv.hoTen);

cout *<<* "Nhap ngay sinh: "; *getline*(is, sv.ngaySinh);

cout *<<* "Nhap gioi tinh: "; *getline*(is, sv.gioiTinh);

cout *<<* "Nhap lop: "; *getline*(is, sv.lopHoc);

cout *<<* "Nhap diem tong ket: "; is *>>* sv.diemTongKet;

*return* is;

}

*friend* ostream*&* *operator<<*(ostream*&* os, *const* SinhVien*&* sv) {

os *<<* "| " *<<* *setw*(10) *<<* sv.maSV

*<<* " | " *<<* *setw*(20) *<<* sv.hoTen

*<<* " | " *<<* *setw*(12) *<<* sv.ngaySinh

*<<* " | " *<<* *setw*(10) *<<* sv.gioiTinh

*<<* " | " *<<* *setw*(10) *<<* sv.lopHoc

*<<* " | " *<<* *setw*(10) *<<* sv.diemTongKet

*<<* " |" *<<* *endl*;

os *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

*return* os;

}

* + - 1. Toán tử so sánh (<)
* Sử dụng để sắp xếp sinh viên theo điểm tổng kết và tên.

bool *operator<*(*const* SinhVien*&* other) *const* {

*if* (diemTongKet != other.diemTongKet) {

*return* diemTongKet > other.diemTongKet;

    }

*return* hoTen *<* other.hoTen;

}

* 1. LỚP “QuanLySinhVien”
     1. Mục đích
* Quản lý danh sách sinh viên, hỗ trợ các thao tác thêm, xoá, sửa, tìm kiếm, sắp xếp và lưu trữ dữ liệu.
  + 1. Thuộc tính
* dsSinhVien (list<SinhVien>): Danh sách chứa nhiều đối tượng SinhVien.
  + 1. Phương thức
* docTuFile(const string& filename): Đọc danh sách sinh viên từ file và thêm vào dsSinhVien.

void *docTuFile*(*const* string*&* filename) {

    ifstream *inFile*(filename);

*if* (*!*inFile) {

        cout *<<* "Khong the mo file!\n";

*return*;

    }

    string line;

*while* (*getline*(inFile, line)) {

        stringstream *ss*(line);

        string ma, ten, ns, gt, lop;

        double dtk;

*getline*(ss, ma, ',');

*getline*(ss, ten, ',');

*getline*(ss, ns, ',');

*getline*(ss, gt, ',');

*getline*(ss, lop, ',');

        ss *>>* dtk;

        dsSinhVien.*emplace\_back*(ma, ten, ns, gt, lop, dtk);

    }

    inFile.*close*();

    cout *<<* "Doc file thanh cong.\n";

}

* ghiVaoFile(const string& filename): Ghi danh sách sinh viên từ dsSinhVien vào file.

void *ghiVaoFile*(*const* string*&* filename) *const* {

    ofstream *outFile*(filename);

*if* (*!*outFile) {

        cout *<<* "Khong the mo file de ghi!\n";

*return*;

    }

*for* (*const* auto& sv : dsSinhVien) {

        outFile *<<* sv.*getMaSV*() *<<* "," *<<* sv.*getHoTen*() *<<* ","

*<<* sv.*getNgaySinh*() *<<* "," *<<* sv.*getGioiTinh*() *<<* ","

*<<* sv.*getLopHoc*() *<<* "," *<<* sv.*getDiemTongKet*() *<<* *endl*;

    }

    outFile.*close*();

    cout *<<* "Ghi file thanh cong.\n";

}

* themSinhVien(const SinhVien& sv): Thêm một sinh viên vào danh sách.

void *themSinhVien*(*const* SinhVien*&* sv) {

    dsSinhVien.*push\_back*(sv);

}

* xoaSinhVien(const string& ma): Xóa sinh viên khỏi danh sách dựa trên mã sinh viên.

void *xoaSinhVien*(*const* string*&* ma) {

    auto it = *remove\_if*(dsSinhVien.*begin*(), dsSinhVien.*end*(),

                        [&](*const* SinhVien*&* sv) { *return* sv.*getMaSV*() *==* ma; });

*if* (it *!=* dsSinhVien.*end*()) {

        dsSinhVien.*erase*(it, dsSinhVien.*end*());

        cout *<<* "Xoa sinh vien thanh cong.\n";

    } *else* {

        cout *<<* "Khong tim thay sinh vien voi ma " *<<* ma *<<* *endl*;

    }

}

* suaSinhVien(const string& ma): Sửa thông tin của sinh viên dựa trên mã sinh viên.

void *suaSinhVien*(*const* string*&* ma) {

*for* (auto& sv : dsSinhVien) {

*if* (sv.*getMaSV*() *==* ma) {

            string ten, ns, gt, lop;

            double dtk;

            cout *<<* "Nhap thong tin moi:\n";

            cin.*ignore*();

            cout *<<* "Ho va Ten: "; *getline*(cin, ten);

            cout *<<* "Ngay Sinh: "; *getline*(cin, ns);

            cout *<<* "Gioi Tinh: "; *getline*(cin, gt);

            cout *<<* "Lop: "; *getline*(cin, lop);

            cout *<<* "Diem Tong Ket: "; cin *>>* dtk;

            sv.*setHoTen*(ten);

            sv.*setNgaySinh*(ns);

            sv.*setGioiTinh*(gt);

            sv.*setLopHoc*(lop);

            sv.*setDiemTongKet*(dtk);

            cout *<<* "Sua thong tin thanh cong.\n";

*return*;

        }

    }

    cout *<<* "Khong tim thay sinh vien voi ma " *<<* ma *<<* *endl*;

}

* hienThiDanhSach() const: Hiển thị danh sách sinh viên.

void *hienThiDanhSach*() *const* {

*if* (dsSinhVien.*empty*()) {

        cout *<<* "Danh sach sinh vien trong.\n";

*return*;

    }

    cout *<<* "\nDanh sach sinh vien:\n";

    cout *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

*for* (*const* auto& sv : dsSinhVien) {

        cout *<<* sv;

    }

}

* timKiemTheoTen(const string& ten) const: Tìm kiếm sinh viên theo tên.

void *timKiemTheoTen*(*const* string*&* ten) *const* {

    bool found = false;

*for* (*const* auto& sv : dsSinhVien) {

*if* (sv.*getHoTen*() *==* ten) {

            cout *<<* "\nSinh vien can tim la:\n";

            cout *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

            cout *<<* sv;

            found = true;

        }

    }

*if* (!found) {

        cout *<<* "Khong tim thay sinh vien voi ten " *<<* ten *<<* *endl*;

    }

}

* thongKeSoLuong() const: Đếm và hiển thị tổng số sinh viên trong danh sách.

void *thongKeSoLuong*() *const* {

    cout *<<* "Tong so sinh vien: " *<<* dsSinhVien.*size*() *<<* *endl*;

}

* sapXepDanhSach(): Sắp xếp danh sách sinh viên theo điểm tổng kết và tên.

void *sapXepDanhSach*() {

    dsSinhVien.*sort*();

    cout *<<* "Da sap xep danh sach sinh vien.\n";

}

* 1. LỚP “App”
     1. Mục đích
* Điều khiển luồng chương trình thông qua một menu điều khiển các chức năng của QuanLySinhVien.
  + 1. Thuộc tính
* qlsv (QuanLySinhVien): Đối tượng quản lý danh sách sinh viên, cung cấp các thao tác quản lý sinh viên.
  + 1. Phương thức
* menu(): Hiển thị menu và gọi các chức năng tương ứng trong lớp QuanLySinhVien theo lựa chọn của người dùng.

void *menu*() {

    int choice;

*do* {

        cout *<<* "\n==== QUAN LY SINH VIEN ====\n";

        cout *<<* "1. Nhap danh sach sinh vien tu file\n";

       cout *<<* "2. Ghi danh sach sinh vien vao file\n";

        cout *<<* "3. Them sinh vien moi\n";

        cout *<<* "4. Xoa sinh vien\n";

        cout *<<* "5. Sua thong tin sinh vien\n";

        cout *<<* "6. Hien thi danh sach sinh vien\n";

        cout *<<* "7. Tim kiem sinh vien theo ten\n";

        cout *<<* "8. Thong ke so luong sinh vien\n";

        cout *<<* "9. Sap xep danh sach sinh vien\n";

        cout *<<* "0. Thoat\n";

        cout *<<* "Lua chon cua ban: ";

        cin *>>* choice;

*switch* (choice) {

*case* 1: {

                string filename;

                cout *<<* "Nhap ten file: ";

                cin *>>* filename;

                qlsv.*docTuFile*(filename);

*break*;

            }

*case* 2: {

                string filename;

                cout *<<* "Nhap ten file: ";

                cin *>>* filename;

                qlsv.*ghiVaoFile*(filename);

*break*;

            }

*case* 3: {

                char choice;

*do* {

                    SinhVien *sv*("", "", "", "", "", 0.0);

                    cin *>>* sv;

                    qlsv.*themSinhVien*(sv);

                    cout *<<* "Ban co muon nhap sinh vien khac khong? (c/k): ";

                    cin *>>* choice;

                } *while* (choice == 'c' || choice == 'C');

*break*;

            }

*case* 4: {

                string ma;

                cout *<<* "Nhap ma sinh vien can xoa: ";

                cin *>>* ma;

                qlsv.*xoaSinhVien*(ma);

*break*;

            }

*case* 5: {

                string ma;

                cout *<<* "Nhap ma sinh vien can sua: ";

                cin *>>* ma;

                qlsv.*suaSinhVien*(ma);

*break*;

            }

*case* 6:

                qlsv.*hienThiDanhSach*();

*break*;

*case* 7: {

                string ten;

                cout *<<* "Nhap ten sinh vien can tim: ";

                cin.*ignore*();

*getline*(cin, ten);

                qlsv.*timKiemTheoTen*(ten);

*break*;

            }

*case* 8:

                qlsv.*thongKeSoLuong*();

*break*;

*case* 9:

                qlsv.*sapXepDanhSach*();

*break*;

*case* 0:

                cout *<<* "Thoat chuong trinh.\n";

*break*;

*default*:

                cout *<<* "Lua chon khong hop le.\n";

        }

    } *while* (choice != 0);

}

1. **Cài đặt các lớp và hàm main bằng C++**

#*include* <iostream>

#*include* <fstream>

#*include* <string>

#*include* <iomanip>

#*include* <list>

#*include* <algorithm>

#*include* <sstream>

using namespace std;

class SinhVien {

private:

    string maSV;

    string hoTen;

    string ngaySinh;

    string gioiTinh;

    string lopHoc;

    double diemTongKet;

public:

*SinhVien*(string ms, string ht, string ns, string gt, string lh, double dtk)

        : *maSV*(ms), *hoTen*(ht), *ngaySinh*(ns), *gioiTinh*(gt), *lopHoc*(lh), *diemTongKet*(dtk) {}

    string *getMaSV*() *const* { *return* maSV; }

    string *getHoTen*() *const* { *return* hoTen; }

    string *getNgaySinh*() *const* { *return* ngaySinh; }

    string *getGioiTinh*() *const* { *return* gioiTinh; }

    string *getLopHoc*() *const* { *return* lopHoc; }

    double *getDiemTongKet*() *const* { *return* diemTongKet; }

    void *setHoTen*(string ht) { hoTen *=* ht; }

    void *setNgaySinh*(string ns) { ngaySinh *=* ns; }

    void *setGioiTinh*(string gt) { gioiTinh *=* gt; }

    void *setLopHoc*(string lh) { lopHoc *=* lh; }

    void *setDiemTongKet*(double dtk) { diemTongKet = dtk; }

*friend* istream*&* *operator>>*(istream*&* is, SinhVien*&* sv) {

        cout *<<* "Nhap ma SV: "; is *>>* sv.maSV; is.*ignore*();

        cout *<<* "Nhap ho va ten: "; *getline*(is, sv.hoTen);

        cout *<<* "Nhap ngay sinh: "; *getline*(is, sv.ngaySinh);

        cout *<<* "Nhap gioi tinh: "; *getline*(is, sv.gioiTinh);

        cout *<<* "Nhap lop: "; *getline*(is, sv.lopHoc);

        cout *<<* "Nhap diem tong ket: "; is *>>* sv.diemTongKet;

*return* is;

    }

*friend* ostream*&* *operator<<*(ostream*&* os, *const* SinhVien*&* sv) {

        os *<<* "| " *<<* *setw*(10) *<<* sv.maSV

*<<* " | " *<<* *setw*(20) *<<* sv.hoTen

*<<* " | " *<<* *setw*(12) *<<* sv.ngaySinh

*<<* " | " *<<* *setw*(10) *<<* sv.gioiTinh

*<<* " | " *<<* *setw*(10) *<<* sv.lopHoc

*<<* " | " *<<* *setw*(10) *<<* sv.diemTongKet

*<<* " |" *<<* *endl*;

        os *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

*return* os;

    }

    bool *operator<*(*const* SinhVien*&* other) *const* {

*if* (diemTongKet != other.diemTongKet) {

*return* diemTongKet > other.diemTongKet;

        }

*return* hoTen *<* other.hoTen;

    }

};

class QuanLySinhVien {

private:

    list<SinhVien> dsSinhVien;

public:

    void *docTuFile*(*const* string*&* filename) {

        ifstream *inFile*(filename);

*if* (*!*inFile) {

            cout *<<* "Khong the mo file!\n";

*return*;

        }

        string line;

*while* (*getline*(inFile, line)) {

            stringstream *ss*(line);

            string ma, ten, ns, gt, lop;

            double dtk;

*getline*(ss, ma, ',');

*getline*(ss, ten, ',');

*getline*(ss, ns, ',');

*getline*(ss, gt, ',');

*getline*(ss, lop, ',');

            ss *>>* dtk;

            dsSinhVien.*emplace\_back*(ma, ten, ns, gt, lop, dtk);

        }

        inFile.*close*();

        cout *<<* "Doc file thanh cong.\n";

    }

    void *ghiVaoFile*(*const* string*&* filename) *const* {

        ofstream *outFile*(filename);

*if* (*!*outFile) {

            cout *<<* "Khong the mo file de ghi!\n";

*return*;

        }

*for* (*const* auto& sv : dsSinhVien) {

            outFile *<<* sv.*getMaSV*() *<<* "," *<<* sv.*getHoTen*() *<<* ","

*<<* sv.*getNgaySinh*() *<<* "," *<<* sv.*getGioiTinh*() *<<* ","

*<<* sv.*getLopHoc*() *<<* "," *<<* sv.*getDiemTongKet*() *<<* *endl*;

        }

        outFile.*close*();

        cout *<<* "Ghi file thanh cong.\n";

    }

    void *themSinhVien*(*const* SinhVien*&* sv) {

        dsSinhVien.*push\_back*(sv);

    }

    void *xoaSinhVien*(*const* string*&* ma) {

        auto it = *remove\_if*(dsSinhVien.*begin*(), dsSinhVien.*end*(),

                            [&](*const* SinhVien*&* sv) { *return* sv.*getMaSV*() *==* ma; });

*if* (it *!=* dsSinhVien.*end*()) {

            dsSinhVien.*erase*(it, dsSinhVien.*end*());

            cout *<<* "Xoa sinh vien thanh cong.\n";

        } *else* {

            cout *<<* "Khong tim thay sinh vien voi ma " *<<* ma *<<* *endl*;

        }

    }

    void *suaSinhVien*(*const* string*&* ma) {

*for* (auto& sv : dsSinhVien) {

*if* (sv.*getMaSV*() *==* ma) {

                string ten, ns, gt, lop;

                double dtk;

                cout *<<* "Nhap thong tin moi:\n";

                cin.*ignore*();

                cout *<<* "Ho va Ten: "; *getline*(cin, ten);

                cout *<<* "Ngay Sinh: "; *getline*(cin, ns);

                cout *<<* "Gioi Tinh: "; *getline*(cin, gt);

                cout *<<* "Lop: "; *getline*(cin, lop);

                cout *<<* "Diem Tong Ket: "; cin *>>* dtk;

                sv.*setHoTen*(ten);

                sv.*setNgaySinh*(ns);

                sv.*setGioiTinh*(gt);

                sv.*setLopHoc*(lop);

                sv.*setDiemTongKet*(dtk);

                cout *<<* "Sua thong tin thanh cong.\n";

*return*;

            }

        }

        cout *<<* "Khong tim thay sinh vien voi ma " *<<* ma *<<* *endl*;

    }

    void *hienThiDanhSach*() *const* {

*if* (dsSinhVien.*empty*()) {

            cout *<<* "Danh sach sinh vien trong.\n";

*return*;

        }

        cout *<<* "\nDanh sach sinh vien:\n";

        cout *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

*for* (*const* auto& sv : dsSinhVien) {

            cout *<<* sv;

        }

    }

    void *timKiemTheoTen*(*const* string*&* ten) *const* {

        bool found = false;

*for* (*const* auto& sv : dsSinhVien) {

*if* (sv.*getHoTen*() *==* ten) {

                cout *<<* "\nSinh vien can tim la:\n";

                cout *<<* "+------------+----------------------+--------------+------------+------------+------------+" *<<* *endl*;

                cout *<<* sv;

                found = true;

            }

        }

*if* (!found) {

            cout *<<* "Khong tim thay sinh vien voi ten " *<<* ten *<<* *endl*;

        }

    }

    void *thongKeSoLuong*() *const* {

        cout *<<* "Tong so sinh vien: " *<<* dsSinhVien.*size*() *<<* *endl*;

    }

    void *sapXepDanhSach*() {

        dsSinhVien.*sort*();

        cout *<<* "Da sap xep danh sach sinh vien.\n";

    }

};

class App {

private:

    QuanLySinhVien qlsv;

public:

    void *menu*() {

        int choice;

*do* {

            cout *<<* "\n==== QUAN LY SINH VIEN ====\n";

            cout *<<* "1. Nhap danh sach sinh vien tu file\n";

            cout *<<* "2. Ghi danh sach sinh vien vao file\n";

            cout *<<* "3. Them sinh vien moi\n";

            cout *<<* "4. Xoa sinh vien\n";

            cout *<<* "5. Sua thong tin sinh vien\n";

            cout *<<* "6. Hien thi danh sach sinh vien\n";

            cout *<<* "7. Tim kiem sinh vien theo ten\n";

            cout *<<* "8. Thong ke so luong sinh vien\n";

            cout *<<* "9. Sap xep danh sach sinh vien\n";

            cout *<<* "0. Thoat\n";

            cout *<<* "Lua chon cua ban: ";

            cin *>>* choice;

*switch* (choice) {

*case* 1: {

                    string filename;

                    cout *<<* "Nhap ten file: ";

                    cin *>>* filename;

                    qlsv.*docTuFile*(filename);

*break*;

                }

*case* 2: {

                    string filename;

                    cout *<<* "Nhap ten file: ";

                    cin *>>* filename;

                    qlsv.*ghiVaoFile*(filename);

*break*;

                }

*case* 3: {

                    char choice;

*do* {

                        SinhVien *sv*("", "", "", "", "", 0.0);

                        cin *>>* sv;

                        qlsv.*themSinhVien*(sv);

                        cout *<<* "Ban co muon nhap sinh vien khac khong? (c/k): ";

                        cin *>>* choice;

                    } *while* (choice == 'c' || choice == 'C');

*break*;

                }

*case* 4: {

                    string ma;

                    cout *<<* "Nhap ma sinh vien can xoa: ";

                    cin *>>* ma;

                    qlsv.*xoaSinhVien*(ma);

*break*;

                }

*case* 5: {

                    string ma;

                    cout *<<* "Nhap ma sinh vien can sua: ";

                    cin *>>* ma;

                    qlsv.*suaSinhVien*(ma);

*break*;

                }

*case* 6:

                    qlsv.*hienThiDanhSach*();

*break*;

*case* 7: {

                    string ten;

                    cout *<<* "Nhap ten sinh vien can tim: ";

                    cin.*ignore*();

*getline*(cin, ten);

                    qlsv.*timKiemTheoTen*(ten);

*break*;

                }

*case* 8:

                    qlsv.*thongKeSoLuong*();

*break*;

*case* 9:

                    qlsv.*sapXepDanhSach*();

*break*;

*case* 0:

                    cout *<<* "Thoat chuong trinh.\n";

*break*;

*default*:

                    cout *<<* "Lua chon khong hop le.\n";

            }

        } *while* (choice != 0);

    }

};

int *main*() {

    App app;

    app.*menu*();

*return* 0;

}

1. **Phân tích thời gian chạy của từng phương thức**
   * + 1. Phương thức docTuFile(const string& filename):

* Độ phức tạp thời gian:
  + Mở và đọc file: Đọc từng dòng của file với getline và mỗi dòng được xử lý một lần.
  + Xử lý mỗi dòng sẽ bao gồm việc phân tích chuỗi thành các phần và tạo một đối tượng SinhVien.
  + Tổng số thao tác đọc dòng sẽ là O(n), với n là số dòng trong file. Mỗi dòng có một độ dài cố định, do đó, việc xử lý mỗi dòng sẽ có độ phức tạp O(1).
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng dòng trong file.
  + - 1. Phương thức ghiVaoFile(const string& filename) const
* Độ phức tạp thời gian:
  + Phương thức này lặp qua từng sinh viên trong danh sách dsSinhVien, thực hiện thao tác ghi từng sinh viên vào file.
  + Mỗi lần ghi vào file là một thao tác có độ phức tạp O(1).
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng sinh viên trong danh sách.
  + - 1. Phương thức themSinhVien(const SinhVien& sv)
* Độ phức tạp thời gian:
  + Phương thức này chỉ thêm một sinh viên vào cuối danh sách. Với push\_back, thời gian thực hiện là O(1).
* Kết luận: Thời gian chạy của phương thức này là O(1).
  + - 1. Phương thức xoaSinhVien(const string& ma)
* Độ phức tạp thời gian:
  + remove\_if sẽ duyệt qua toàn bộ danh sách sinh viên để tìm kiếm sinh viên có mã phù hợp, có độ phức tạp là O(n).
  + Sau đó, erase sẽ xóa phần tử trong danh sách, thời gian xóa có độ phức tạp O(k), với k là số lượng phần tử cần xóa.
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng sinh viên trong danh sách.
  + - 1. Phương thức suaSinhVien(const string& ma)
* Độ phức tạp thời gian:
  + Phương thức này sẽ duyệt qua từng sinh viên trong danh sách dsSinhVien để tìm sinh viên có mã tương ứng.
  + Việc duyệt danh sách này có độ phức tạp là O(n).
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng sinh viên trong danh sách.
  + - 1. Phương thức hienThiDanhSach() const
* Độ phức tạp thời gian:
  + Phương thức này chỉ duyệt qua từng sinh viên trong danh sách và in thông tin của từng sinh viên. Việc duyệt qua danh sách có độ phức tạp là O(n).
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng sinh viên trong danh sách.
  + - 1. Phương thức timKiemTheoTen(const string& ten) const
* Độ phức tạp thời gian:
  + Phương thức này duyệt qua tất cả sinh viên trong danh sách để tìm sinh viên có tên tương ứng. Việc duyệt qua danh sách có độ phức tạp là O(n).
* Kết luận: Thời gian chạy của phương thức này là O(n), với n là số lượng sinh viên trong danh sách.
  + - 1. Phương thức thongKeSoLuong() const
* Độ phức tạp thời gian:
  + Phương thức này chỉ đơn giản là gọi hàm size() của list, có độ phức tạp là O(1).
* Kết luận: Thời gian chạy của phương thức này là O(1).
  + - 1. Phương thức sapXepDanhSach()
* Độ phức tạp thời gian:
  + Phương thức này gọi sort trên list, đây là một thuật toán sắp xếp có độ phức tạp O(n log n), với n là số lượng sinh viên trong danh sách.
* Kết luận: Thời gian chạy của phương thức này là O(n log n), với n là số lượng sinh viên trong danh sách.

# **BÀI B**

1. **Đề bài**
   * + 1. Cài đặt cấu dữ liệu trừu tượng cây tìm kiếm nhị phân sử dụng danh sách liên kết.
       2. Ứng dụng cây tìm kiếm nhị phân xây dựng lớp quản lý các đối tượng sinh viên, mỗi sinh viên gồm các thông tin: MSV (MSV là duy nhất), Họ tên, tuổi. Lớp có các phương thức:
          1. Đọc danh sách sinh viên từ tệp vào cây *(cấu trúc file lưu trữ danh sách sinh viên do người lập trình tự định nghĩa)*
          2. Thêm 1 sinh viên vào cây, nếu sinh viên đã tồn tại thì thông báo lỗi.
          3. Xóa 1 sinh viên trong lớp theo MSV. Nếu mã không có thì thông báo lỗi.
          4. Cập nhật thông tin 1 sinh viên khi biết MSV và các thông tin cần cập nhật.
          5. Tìm kiếm sinh viên theo MSV
          6. Hiển thị danh sách sinh viên lên màn hình.
          7. Ghi toàn bộ sinh viên của cây vào file.
2. **Phân tích bài toán**
3. Xác định các yêu cầu của bài toán
   1. Cài đặt cấu trúc dữ liệu cây tìm kiếm nhị phân (Binary Search Tree)

* Mục tiêu: Xây dựng một lớp cài đặt cây tìm kiếm nhị phân với danh sách liên kết, với các thao tác cơ bản như thêm, xóa, tìm kiếm, và duyệt cây.
* Yêu cầu:
  + Cấu trúc cây: Cây tìm kiếm nhị phân (BST) sẽ được cài đặt dưới dạng danh sách liên kết, với mỗi nút có tối đa hai con (con trái và con phải).
  + Các phương thức cơ bản:
    - Thêm phần tử: Thêm một nút mới vào cây theo nguyên tắc của BST (phần tử nhỏ hơn vào bên trái, lớn hơn vào bên phải).
    - Xóa phần tử: Xóa một nút dựa vào giá trị của nó theo cách xóa nút trong BST (bao gồm các trường hợp: nút là lá, nút có một con, và nút có hai con).
    - Tìm kiếm: Tìm kiếm nút có giá trị tương ứng trong BST.
    - Duyệt cây: Các kiểu duyệt cây như in-order, pre-order, post-order để lấy ra danh sách phần tử theo các thứ tự khác nhau.
  1. Ứng dụng cây tìm kiếm nhị phân để quản lý sinh viên
* Mục tiêu: Sử dụng cấu trúc cây tìm kiếm nhị phân đã cài đặt ở Phần 1 để xây dựng một lớp quản lý các đối tượng sinh viên với các phương thức quản lý dữ liệu sinh viên.
* Yêu cầu:
  + Dữ liệu sinh viên: Mỗi sinh viên có các thông tin gồm:
    - MSV: Mã số sinh viên (duy nhất).
    - Họ tên: Tên đầy đủ của sinh viên.
    - Tuổi: Tuổi của sinh viên.
  + Các phương thức:
    - Đọc danh sách sinh viên từ tệp vào cây: Đọc dữ liệu sinh viên từ tệp, xây dựng các nút sinh viên và thêm vào cây. Cấu trúc của file lưu trữ danh sách sinh viên có thể do bạn tự định nghĩa (ví dụ: mỗi dòng chứa thông tin một sinh viên, các thông tin cách nhau bởi dấu phẩy).
    - Thêm sinh viên vào cây: Thêm sinh viên mới vào cây dựa trên MSV. Nếu MSV đã tồn tại trong cây, hiển thị thông báo lỗi.
    - Xóa sinh viên: Xóa sinh viên khỏi cây theo MSV. Nếu MSV không tồn tại, hiển thị thông báo lỗi.
    - Cập nhật thông tin sinh viên: Tìm sinh viên theo MSV, cập nhật thông tin của sinh viên (ví dụ: họ tên hoặc tuổi).
    - Tìm kiếm sinh viên: Tìm kiếm sinh viên trong cây theo MSV.
    - Hiển thị danh sách sinh viên: Hiển thị danh sách sinh viên lên màn hình theo thứ tự in-order.
    - Ghi danh sách sinh viên vào tệp: Lưu toàn bộ danh sách sinh viên từ cây vào tệp (cấu trúc tệp có thể tương tự file đọc vào ban đầu).

1. Xác định các lớp, các thuộc tính, các phương thức của lớp.
   1. File “BinarySearchTree.cpp”
      1. Lớp Node

* Mục đích: đại diện cho một nút trong cây nhị phân, chứa thông tin về dữ liệu và các con trỏ đến nút con bên trái và bên phải.
* Thuộc tính:
  + data: Dữ liệu lưu trữ trong nút, kiểu T (template type).
  + left: Con trỏ đến nút con bên trái.
  + right: Con trỏ đến nút con bên phải.
* Phương thức:
  + Node(T value): Hàm tạo để khởi tạo nút với giá trị value cho data, và con trỏ left, right được gán là nullptr.
    1. Lớp BinarySearchTree
* Mục đích: đại diện cho cây nhị phân tìm kiếm, nơi các thao tác chính của cây được cài đặt như thêm, tìm kiếm, duyệt cây, xóa nút, và các thao tác bổ sung như tính chiều cao và số lượng nút.
* Thuộc tính:
  + root: Con trỏ đến nút gốc của cây.
* Phương thức:
  + BinarySearchTree(): Hàm tạo để khởi tạo cây với gốc là nullptr.
  + ~BinarySearchTree(): Hàm hủy để xóa tất cả các nút trong cây khi cây bị hủy.
  + them(T value): Thêm một nút với giá trị value vào cây.
  + timKiem(T value): Tìm một nút có giá trị value trong cây.
  + duyetGiua(): Duyệt cây theo thứ tự giữa (in-order) và in ra các giá trị.
  + duyetTruoc(): Duyệt cây theo thứ tự trước (pre-order) và in ra các giá trị.
  + duyetSau(): Duyệt cây theo thứ tự sau (post-order) và in ra các giá trị.
  + xoa(T value): Xóa nút có giá trị value khỏi cây.
  + laRong(): Kiểm tra xem cây có rỗng không.
  + xoaCay(): Xóa toàn bộ cây.
  + demNode(): Đếm số lượng nút trong cây.
  + chieuCao(): Tính chiều cao của cây.
* Các phương thức riêng:
  + themNode(Node<T>\* node, T value): Phương thức đệ quy thêm nút vào cây.
  + timNode(Node<T>\* node, T value): Phương thức đệ quy tìm nút trong cây.
  + duyetGiuaNode(Node<T>\* node): Duyệt cây theo thứ tự giữa đệ quy.
  + duyetTruocNode(Node<T>\* node): Duyệt cây theo thứ tự trước đệ quy.
  + duyetSauNode(Node<T>\* node): Duyệt cây theo thứ tự sau đệ quy.
  + timMin(Node<T>\* node): Tìm nút có giá trị nhỏ nhất trong cây con.
  + xoaNode(Node<T>\* node, T value): Xóa một nút có giá trị value.
  + xoaToanBo(Node<T>\* node): Xóa toàn bộ cây đệ quy.
  + demNodeHelper(Node<T>\* node): Đếm số lượng nút trong cây đệ quy.
  + chieuCaoHelper(Node<T>\* node): Tính chiều cao của cây đệ quy.
  1. File “StudentManager.cpp”
     1. Lớp SinhVien (Struct)
* Thuộc tính:
  + msv (string): Mã số sinh viên.
  + ten (string): Tên sinh viên.
  + tuoi (int): Tuổi sinh viên.
* Phương thức:
  + operator<, operator>, operator==: So sánh sinh viên theo mã số sinh viên (MSV).
  + operator<<: In thông tin sinh viên ra màn hình theo định dạng bảng.
    1. Lớp QuanLySinhVien
* Thuộc tính: bst (BinarySearchTree<SinhVien>): Cây nhị phân lưu trữ dữ liệu sinh viên.
* Phương thức:
  + docTuFile(const string& filename): Đọc danh sách sinh viên từ tệp và thêm vào cây.
  + themSinhVien(const SinhVien& sv): Thêm một sinh viên vào cây.
  + xoaSinhVien(const string& msv): Xóa một sinh viên theo MSV.
  + capNhatSinhVien(const string& msv, const string& tenMoi, int tuoiMoi): Cập nhật thông tin sinh viên theo MSV.
  + timSinhVien(const string& msv): Tìm kiếm sinh viên theo MSV và hiển thị thông tin nếu tìm thấy.
  + hienThiSinhVien(): Hiển thị danh sách tất cả sinh viên trong cây.
  + luuVaoFile(const string& filename): Lưu danh sách sinh viên vào tệp.
* Phương thức phụ:
  + ghiVaoFile(Node<SinhVien>\* node, ofstream& file): Ghi thông tin sinh viên từ cây nhị phân vào tệp.
    1. Lớp App
* Thuộc tính:
  + qlsv (QuanLySinhVien): Đối tượng quản lý sinh viên.
* Phương thức:
  + hienThiMenu(): Hiển thị menu cho người dùng chọn các chức năng.
  + menu(): Quản lý vòng lặp menu, gọi các phương thức của QuanLySinhVien dựa trên lựa chọn của người dùng.

1. **Cài đặt các lớp và hàm main bằng C++**

#*include* "BinarySearchTree.cpp"

#*include* <iostream>

#*include* <iomanip>

#*include* <fstream>

#*include* <string>

#*include* <sstream>

using namespace std;

struct SinhVien {

    string msv;

    string ten;

    int tuoi;

    bool *operator<*(*const* SinhVien*&* other) *const* { *return* msv *<* other.msv; }

    bool *operator>*(*const* SinhVien*&* other) *const* { *return* msv *>* other.msv; }

    bool *operator==*(*const* SinhVien*&* other) *const* { *return* msv *==* other.msv; }

*friend* ostream*&* *operator<<*(ostream*&* os, *const* SinhVien*&* sv) {

        os *<<* "| " *<<* *setw*(10) *<<* sv.msv

*<<* " | " *<<* *setw*(20) *<<* sv.ten

*<<* " | " *<<* *setw*(10) *<<* sv.tuoi

*<<* " |" *<<* *endl*;

        os *<<* "+------------+----------------------+------------+" *<<* *endl*;

*return* os;

    }

};

class QuanLySinhVien {

private:

    BinarySearchTree<SinhVien> bst;

public:

// *a. Đọc danh sách sinh viên từ tệp vào cây*

    void *docTuFile*(*const* string*&* filename) {

        ifstream *inFile*(filename);

*if* (*!*inFile) {

            cout *<<* "Khong the mo file!\n";

*return*;

        }

        string line;

*while* (*getline*(inFile, line)) {

            stringstream *ss*(line);

            string msv, ten, tuoi\_str;

            int tuoi;

// *Đọc từng phần dữ liệu từ file*

*getline*(ss, msv, ',');// *Đọc MSV (string)*

*getline*(ss, ten, ',');// *Đọc tên (string)*

*getline*(ss, tuoi\_str, ',');// *Đọc tuổi (string) để chuyển thành int*

*stringstream*(tuoi\_str) *>>* tuoi;

// *Thêm sinh viên vào cây*

            bst.*them*(SinhVien{msv, ten, tuoi});

        }

        inFile.*close*();

        cout *<<* "Doc file thanh cong.\n";

    }

// *b. Thêm 1 sinh viên vào cây*

    void *themSinhVien*(*const* SinhVien*&* sv) {

*if* (bst.*timKiem*(sv)) {

            cout *<<* "Loi: Sinh vien voi MSV " *<<* sv.msv *<<* " da ton tai." *<<* *endl*;

        } *else* {

            bst.*them*(sv);

            cout *<<* "Them sinh vien thanh cong.\n";

        }

    }

// *c. Xóa 1 sinh viên theo MSV*

    void *xoaSinhVien*(*const* string*&* msv) {

        SinhVien sv{msv, "", 0};

*if* (!bst.*timKiem*(sv)) {

            cout *<<* "Loi: Sinh vien voi MSV " *<<* msv *<<* " khong ton tai." *<<* *endl*;

        } *else* {

            bst.*xoa*(sv);

            cout *<<* "Xoa sinh vien thanh cong.\n";

        }

    }

// *d. Cập nhật thông tin 1 sinh viên khi biết MSV*

    void *capNhatSinhVien*(*const* string*&* msv, *const* string*&* tenMoi, int tuoiMoi) {

*xoaSinhVien*(msv);

*themSinhVien*(SinhVien{msv, tenMoi, tuoiMoi});

        cout *<<* "Cap nhat thong tin sinh vien thanh cong.\n";

    }

// *e. Tìm kiếm sinh viên theo MSV*

    void *timSinhVien*(*const* string*&* msv) {

        SinhVien sv{msv, "", 0};// *Tạo đối tượng sinh viên với MSV cần tìm*

        Node<SinhVien>\* node = bst.*timKiem*(sv);// *Tìm kiếm node chứa sinh viên*

*if* (node != nullptr) {// *Nếu tìm thấy node chứa sinh viên*

            cout *<<* "Da tim thay sinh vien: " *<<* *endl*;

            cout *<<* "+------------+----------------------+------------+" *<<* *endl*;

            cout *<<* node->data;// *Hiển thị thông tin sinh viên từ node*

        } *else* {

            cout *<<* "Sinh vien voi MSV " *<<* msv *<<* " khong ton tai." *<<* *endl*;

        }

    }

// *f. Hiển thị danh sách sinh viên lên màn hình*

    void *hienThiSinhVien*() {

        cout *<<* "Danh sach sinh vien:\n";

        cout *<<* "+------------+----------------------+------------+" *<<* *endl*;

        bst.*duyetGiua*();

    }

// *g. Ghi toàn bộ sinh viên của cây vào file*

    void *luuVaoFile*(*const* string*&* filename) {

        ofstream *file*(filename);

*if* (file.*is\_open*()) {

*ghiVaoFile*(bst.root, file);

            file.*close*();

            cout *<<* "Du lieu da duoc luu vao tep.\n";

        } *else* {

            cerr *<<* "Khong the mo tep: " *<<* filename *<<* *endl*;

        }

    }

private:

    void *ghiVaoFile*(Node<SinhVien>*\** node, ofstream*&* file) {

*if* (node != nullptr) {

*ghiVaoFile*(node->left, file);

            file *<<* node->*data*.msv *<<* "," *<<* node->*data*.ten *<<* "," *<<* node->*data*.tuoi *<<* *endl*;

*ghiVaoFile*(node->right, file);

        }

    }

};

class App {

private:

    QuanLySinhVien qlsv;

public:

    void *hienThiMenu*() {

        cout *<<* "\n==== QUAN LY SINH VIEN ====\n";

        cout *<<* "1. Tai danh sach sinh vien tu tep\n";

        cout *<<* "2. Them sinh vien\n";

        cout *<<* "3. Xoa sinh vien\n";

        cout *<<* "4. Cap nhat thong tin sinh vien\n";

        cout *<<* "5. Tim kiem sinh vien\n";

        cout *<<* "6. Hien thi danh sach sinh vien\n";

        cout *<<* "7. Luu danh sach sinh vien vao tep\n";

        cout *<<* "0. Thoat\n";

        cout *<<* "Chon chuc nang: ";

    }

    void *menu*() {

        int luaChon;

*do* {

*hienThiMenu*();

            cin *>>* luaChon;

*switch* (luaChon) {

*case* 1: {

                    string filename;

                    cout *<<* "Nhap ten tep de tai danh sach sinh vien: ";

                    cin *>>* filename;

                    qlsv.*docTuFile*(filename);

*break*;

                }

*case* 2: {

                    string msv, ten;

                    int tuoi;

                    cout *<<* "Nhap MSV sinh vien: ";

                    cin *>>* msv;

                    cout *<<* "Nhap ten sinh vien: ";

                    cin.*ignore*();

*getline*(cin, ten);

                    cout *<<* "Nhap tuoi sinh vien: ";

                    cin *>>* tuoi;

                    qlsv.*themSinhVien*({msv, ten, tuoi});

*break*;

                }

*case* 3: {

                    string msv;

                    cout *<<* "Nhap MSV cua sinh vien can xoa: ";

                    cin *>>* msv;

                    qlsv.*xoaSinhVien*(msv);

*break*;

                }

*case* 4: {

                    string msv, ten;

                    int tuoi;

                    cout *<<* "Nhap MSV cua sinh vien can cap nhat: ";

                    cin *>>* msv;

                    cout *<<* "Nhap ten moi: ";

                    cin.*ignore*();

*getline*(cin, ten);

                    cout *<<* "Nhap tuoi moi: ";

                    cin *>>* tuoi;

                    qlsv.*capNhatSinhVien*(msv, ten, tuoi);

*break*;

                }

*case* 5: {

                    string msv;

                    cout *<<* "Nhap MSV can tim: ";

                    cin *>>* msv;

                    qlsv.*timSinhVien*(msv);

*break*;

                }

*case* 6:

                    qlsv.*hienThiSinhVien*();

*break*;

*case* 7: {

                    string filename;

                    cout *<<* "Nhap ten tep de luu danh sach sinh vien: ";

                    cin *>>* filename;

                    qlsv.*luuVaoFile*(filename);

*break*;

                }

*case* 0:

                    cout *<<* "Thoat chuong trinh.\n";

*break*;

*default*:

                    cout *<<* "Lua chon khong hop le. Vui long chon lai.\n";

*break*;

            }

        } *while* (luaChon != 0);

    }

};

int *main*() {

    App app;

    app.*menu*();

*return* 0;

}

1. **Phân tích thời gian chạy của từng phương thức**
2. File “BinarySearchTree.cpp”
   1. them(T value):

* Thời gian chạy: O(n), trong đó n là chiều cao của cây.
* Giải thích: Phương thức này thêm một phần tử vào cây. Việc thêm vào cây tìm kiếm nhị phân yêu cầu đi qua các nút theo chiều cao của cây. Do đó, thời gian chạy là O(n).
  1. timKiem(T value):
* Thời gian chạy: O(h), trong đó h là chiều cao của cây.
* Giải thích: Phương thức tìm kiếm yêu cầu duyệt cây theo chiều sâu (depth-first search). Trong trường hợp xấu nhất, việc tìm kiếm sẽ phải đi qua tất cả các nút trên một nhánh của cây, dẫn đến thời gian chạy O(h).
  1. duyetGiua() (In-order traversal):
* Thời gian chạy: O(h), trong đó h là số lượng nút trong cây.
* Giải thích: Phương thức này duyệt qua tất cả các nút của cây, thực hiện một lần duyệt đệ quy với mỗi nút, vì vậy thời gian chạy là O(h).
  1. duyetTruoc() (Pre-order traversal):
* Thời gian chạy: O(n), trong đó n là số lượng nút trong cây.
* Giải thích: Tương tự như phương thức duyetGiua(), phương thức này duyệt qua tất cả các nút của cây, do đó thời gian chạy là O(n).
  1. duyetSau() (Post-order traversal):
* Thời gian chạy: O(n), trong đó n là số lượng nút trong cây.
* Giải thích: Đây là một phương thức duyệt cây theo kiểu đệ quy và sẽ thăm tất cả các nút trong cây, vì vậy thời gian chạy là O(n).
  1. xoa(T value):
* Thời gian chạy: O(h), trong đó h là chiều cao của cây.
* Giải thích: Việc xóa một nút trong cây tìm kiếm nhị phân yêu cầu duyệt cây để tìm nút cần xóa, sau đó điều chỉnh lại cấu trúc cây. Tối đa phải duyệt qua h nút, vì vậy thời gian chạy là O(h).
  1. laRong():
* Thời gian chạy: O(1).
* Giải thích: Phương thức này chỉ kiểm tra xem cây có rỗng hay không bằng cách kiểm tra xem root có phải là nullptr. Thời gian chạy là hằng số, O(1).
  1. xoaCay():
* Thời gian chạy: O(n), trong đó n là số lượng nút trong cây.
* Giải thích: Phương thức này xóa tất cả các nút trong cây. Nó sẽ duyệt qua tất cả các nút của cây và xóa chúng. Do đó, thời gian chạy là O(n).
  1. demNode():
* Thời gian chạy: O(n), trong đó n là số lượng nút trong cây.
* Giải thích: Phương thức này đếm số lượng nút trong cây bằng cách duyệt qua tất cả các nút, vì vậy thời gian chạy là O(n).
  1. chieuCao():
* Thời gian chạy: O(n), trong đó n là số lượng nút trong cây.
* Giải thích: Phương thức này tính chiều cao của cây bằng cách duyệt qua tất cả các nút trong cây, vì vậy thời gian chạy là O(n).
  1. Kết luận
* Các phương thức trong cây tìm kiếm nhị phân sẽ có thời gian chạy O(h) trong trường hợp tốt nhất và O(n) trong trường hợp xấu nhất. Nếu cây có dạng cân bằng (như AVL hoặc Red-Black tree), chiều cao h có thể là O(log n), và trong trường hợp này, thời gian chạy của các phương thức là O(log n). Tuy nhiên, trong trường hợp cây có dạng chuỗi (một nhánh dài), chiều cao h có thể là O(n), dẫn đến thời gian chạy trong trường hợp xấu nhất là O(n).

1. File “StudentManagement.cpp”
   1. docTuFile(const string& filename):

* Thời gian chạy là O(n), trong đó n là số lượng dòng trong tệp. Mỗi dòng chứa thông tin của một sinh viên, và thời gian đọc và xử lý mỗi dòng là hằng số. Sau khi đọc mỗi dòng, bạn thêm một sinh viên vào cây nhị phân, với độ phức tạp là O(h) cho mỗi lần thêm. Tổng thời gian là O(n \* h), với h là chiều cao của cây. Nếu cây là cân bằng, thời gian chạy sẽ là O(n \* log n).
  1. themSinhVien(const SinhVien& sv):
* Thời gian chạy là O(h), vì cây sẽ thực hiện tìm kiếm (O(h)) và thêm một node (O(h)).
  1. xoaSinhVien(const string& msv):
* Thời gian chạy là O(h), vì cây sẽ tìm kiếm sinh viên cần xóa và sau đó xóa nó, mỗi thao tác tốn thời gian O(h).
  1. capNhatSinhVien(const string& msv, const string& tenMoi, int tuoiMoi):
* Thời gian chạy là O(h) cho việc xóa và O(h) cho việc thêm lại sinh viên, tổng cộng là O(h).
  1. timSinhVien(const string& msv):
* Thời gian chạy là O(h), vì bạn cần tìm kiếm sinh viên trong cây, và thời gian tìm kiếm phụ thuộc vào chiều cao cây.
  1. hienThiSinhVien():
* Thời gian chạy là O(n), vì bạn phải duyệt toàn bộ cây để in thông tin sinh viên.
  1. luuVaoFile(const string& filename):
* Thời gian chạy là O(n), vì bạn cần duyệt qua toàn bộ cây và ghi thông tin mỗi sinh viên vào tệp.

**DANH SÁCH TÀI LIỆU THAM KHẢO**

[1] https://www.w3schools.com/cpp/cpp\_oop.asp

[2] https://en.cppreference.com/w/cpp/container/list

[3] https://www.geeksforgeeks.org/cpp-binary-search-tree/

[4] https://codeofcode.org/lessons/binary-search-trees-in-cpp/

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với thầy Phạm Xuân Tích đã nhiệt tình hướng dẫn và giảng dạy để em hoàn thành tốt báo cáo bài tập lớn.

Trong quá trình học tập và làm bài báo cáo bài tập lớn, do kiến thức còn nhiều hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy để em học hỏi được nhiều kĩ năng, kinh nghiệm và sẽ hoàn thành tốt hơn trong những bài báo cáo sau này.

Em xin chân thành cảm ơn!