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 Văn Tú - 221231028

Lớp CNTT 2 - K63

Đề tài: A: Quản lý thư viện

B: Bài 23.

# LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện em đã được trang bị các kiến thức, các kỹ năng thực tế để có thể hoàn thành bài tập lớn môn học của mình.

Em xin gửi lời cảm ơn chân thành đến thầy Phạm Xuân Tích đã quan tâm, hướng dẫn, truyền đạt những kiến thức và kinh nghiệm cho em và các bạn trong suốt thời gian học tập môn Cấu trúc dữ liệu và giải thuật.

Trong quá trình làm bài tập lớn không tránh khỏi được những sai sót, em mong nhận được sự góp ý của thầy và các bạn để được hoàn thiện hơn.

TP Hà Nội, tháng 11, năm 2023

# Mục Lục

# Phần A: Chương trình quản lý thư viện bằng vector .

❖ **Lý do chọn đề tài:**

Trong thời đại tri thức ngày nay, việc nâng cao chất lượng giáo dục là nhiệm vụ quan trọng và hàng đầu của nước ta. Song song với việc đào tạo, việc quản lý cũng không kém phần quan trọng đặc biệt là việc quản lý sách trong các thư viện . Hằng ngày một số lượng lớn sách trong các thư viện được sử dụng. Việc quản lý sách vốn dĩ đã khá khó khăn nhưng do nhu cầu đọc của chúng ta mỗi ngày càng tăng nên việc quản lý sách, các phiếu mượn sách cũng như quản lý sinh viên, bạn đọc ra vào trong các thư viện càng khó khăn hơn.

Xuất phát từ nhu cầu thiết thực đó việc tạo ra chương trình Quản lý thư viện như một phần tất yếu, giúp giải quyết phần nào khó khăn đặt ra ở trên, hỗ trợ các chức năng thông dụng mà nhiệm vụ của một thủ thư hay làm và việc lưu trữ dữ liệu sách,...

## I. Đề bài

Hệ thống quản lý thư viện ứng dụng bằng ngôn ngữ C++ sử dụng vector.

– Chương trình bắt đầu với một giao diện đăng nhập. Người dùng nhập tên đăng nhập và mật khẩu.

Khi đăng nhập với tài khoản Admin chương trình chạy đến giao diện Admin. Khi đăng nhập với tài khoản người dùng đã đăng kí trong thư viện thì sẽ chạy đến giao diện User, có chức năng đăng ký tài khoản mới và log out khi cần.

– Thư viện cần quản lí 3 loại thông tin gồm sách, độc giả và các phiếu mượn/trả sách.

Thông tin thẻ độc giả cần quản lí bao gồm:mã sinh viên, tên, ngành học, khoá, ngày sinh.

Thông tin sách cần quản lí bao gồm:mã sách, tên sách,thể loại, tác giả, năm xuất bản, giá sách, số quyển sách.

Mỗi phiếu mượn/trả sách chứa thông tin về mã độc giả,mã sách mượn, tên người mượn, ngày mượn, ngày trả. Mỗi sách được mượn tối đa trong 7 ngày, nếu quá hạn sẽ bị cảnh báo.

- Viết chương trình có các chức năng sau:

Admin:

1. Quản lý sách:

- Thêm(đầu, giữa, cuối), sửa, xoá sách.

- In danh sách sách có trong thư viện.

2. Quản lý phiếu mượn trả sách của sinh viên:

- Mượn, trả sách (giới hạn mỗi phiếu mượn tối đa 7 ngày).

- In danh sách các phiếu mượn sách của thư viện, các phiếu mượn quá hạn.

3. Quản lý sinh viên bạn đọc ra vào thư viện:

- Thêm, sửa, xoá sinh viên.

- In danh sách sinh viên trong thư viện.

4. Sắp xếp:

- Sắp xếp sách theo giá tiền tăng dần, thứ tự alphabet.

- Sắp xếp sinh viên theo thứ tự alphabet tăng dần theo tên + họ.

5. Tìm kiếm:

- Tìm kiếm sách theo mã sách, tên sách.

- Tìm kiếm sinh viên theo mã sinh viên, tên sinh viên.

- Tìm kiếm phiếu sách theo tên sinh viên.

6.Thống kê:

-Thống kê số sinh viên, đầu sách, phiếu mượn trong thư viện.

User:

1 In toàn bộ sách có trong thư viện.

2 Mượn sách.

3 Trả sách.

4 In ra các phiếu mượn sách của bạn.

5 Tìm kiếm sách theo tên.

6 Sắp xếp sách theo thứ tự tăng dần giá tiền.

## II. Phân tích bài toán:

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

-Sử dụng vector để lưu trữ danh sách sách, sinh viên, phiếu mượn, và các thông tin liên quan

=> Tại vấn đề này em sẽ xây dựng 7 class gồm thời gian, sách, bạn đọc, phiếu mượn trả sách, quản lý sách, quản lý bạn đọc và quản lý thư viện thêm class ứng dụng và  class làm đẹp nữa là 9.

- Sử dụng file text để lưu trữ và đọc dữ liệu đồng bộ với vector, đảm bảo tính liên tục của dữ liệu giữa các lần chạy chương trình.

-Sử dụng các hàm để thực hiện các chức năng cụ thể của từng menu.

- Bảo đảm rằng chương trình có thể xử lý các trường hợp ngoại lệ và thông báo lỗi khi cần thiết.

- Chương trình cho phép lưu các danh sách vào file (lưu trữ và đọc dữ liệu) và đồng bộ file với vector; Kiểm tra các điều kiện khi nhập làm dữ liệu bị sai.

- Xây dựng lớp đối tượng quản lý có ít nhất 4 trường dữ liệu với ít nhất 3 toán tử gồm toán tử nhập >>, xuất << và toán tử so sánh <.

- Xây dựng lớp danh sách đối tượng quản lý cho phép nhập, xuất, sắp xếp danh sách quản lý và ít nhất 3 thao tác.

- Xây dựng lớp app để quản lý có menu và thực hiện các thao tác ở lớp danh sách.

- Chương trình ban đầu có phân quyền giữa Admin và User.

=> Thêm 1 class login_signup. Vậy có 10 class tất cả.

Em tổ chức các chức năng:

Admin:

1.Quản lý sách:

a. Thêm sách vào thư viện (chọn chức năng thêm vào đầu, giữa hoặc cuối rồi nhập vào mã sách, tên sách, thể loại, tác giả, năm xuất bản, giá tiền, số lượng. Kiểm tra các điều kiện nhập, có trùng thông tin với các quyển khác trước đó hay không, mặc định thêm vào file Sach.txt và vector.

b. Sửa thông tin sách qua mã sách [ nhập vào mã sách chương trình sẽ tự động xuất ra thông tin của quyển sách (nếu mã đúng) ].

c. Xoá một quyển sách qua mã sách (chú ý muốn xoá thì sách trong thư viện phải cần đủ số lượng ban đầu vì nếu có người đang mượn thì không xoá được).

d. In toàn bộ sách trong thư viện (đọc dữ liệu từ vector và liệt kê các thông tin sách dưới dạng bảng)

2. Quản lý Phiếu mượn sách của độc giả:

a. Thêm phiếu mượn sách và đánh mã tự động(nhập vào mã thẻ sinh viên, họ tên, mã sách nếu validation thành công chương trình sẽ in ra phiếu mượn sách có thời gian ngày tháng bạn mượn sách và lưu vào file `PhieuMuon.txt` với vector, lưu ý mỗi sinh viên chỉ được mượn tối đa 7 ngày)

b. Trả sách (nhập vào nhập vào mã thẻ sinh viên, chương trình sẽ xuất ra tất cả các cuốn sách mà bạn đang mượn, rồi bạn nhập mã sách muốn trả và họ tên để hoàn tất thủ tục.

c. In danh sách các phiếu mượn sách (đọc dữ liệu từ vector và liệt kê các thông tin phiếu mượn dưới dạng bảng)

d. In danh sách các phiếu mượn sách quá hạn (duyệt qua tất cả các phiếu mượn sách nếu phiếu nào có thời gian ngày trả nhỏ hơn so với thời gian thực thì in ra).

3.Quản lý sinh viên ra vào thư viên: (chức năng thêm, sửa, xoá, in tương tự như  sách)

a. Thêm một sinh viên vào thư viện.(Nhập vào mã sinh viên, tên, ngành học, khoá, ngày sinh)

b. Sửa thông tin sinh viên qua mã sinh viên.

c. Xoá thông tin sinh viên ra khỏi thư viện qua mã sinh viên.

d. In danh sách sinh viên đang ở trong thư viện.

4. Sắp xếp: (dùng hàm sort có sẵn trong thư viện)

a. Sắp xếp sách theo giá tiền tăng dần.

b. Sắp xếp sách theo theo thứ tự alphabet.

c. Sắp xếp danh sách các bạn đọc theo thứ tự alphabet.

5. Tìm kiếm: (tìm kiếm tuyến tính, tìm kiếm thông tin bạn đọc, sách, phiếu mượn rồi xuất ra dưới dạng bảng)

a. Tìm kiếm sách theo mã sách.

b. Tìm kiếm sách theo name.

c. Tìm kiếm sinh viên trong thư viện theo mã sinh viên.

d. Tìm kiếm sinh viên trong thư viện theo name.

e. Tìm kiếm phiếu sách theo tên sinh viên.

6. Thống kê:

a. Thống kê số sinh viên trong thư viện

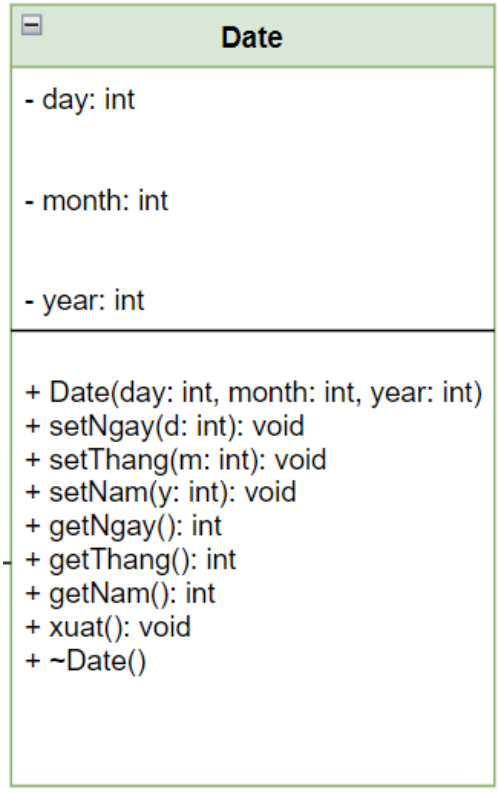
b. Thống kê số đầu sách trong thư viện

c. Thống kê số phiếu mượn sách

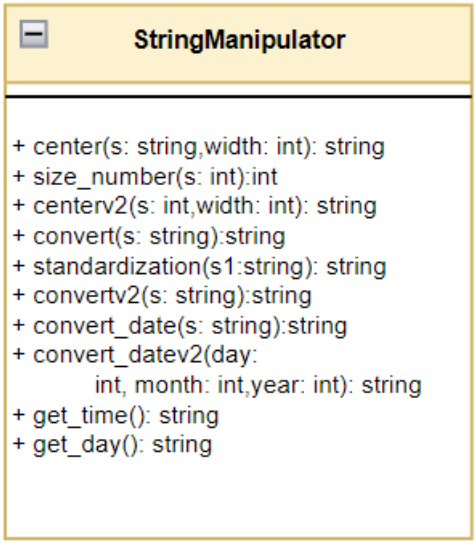
User: (Các hàm dùng đều tương tự như trên)

1. In toàn bộ sách có trong thư viện.
2. Mượn sách.
3. Trả sách.
4. In ra các phiếu mượn sách của bạn.
5. Tìm kiếm sách theo tên.
6. Sắp xếp sách theo thứ tự tăng dần giá tiền.

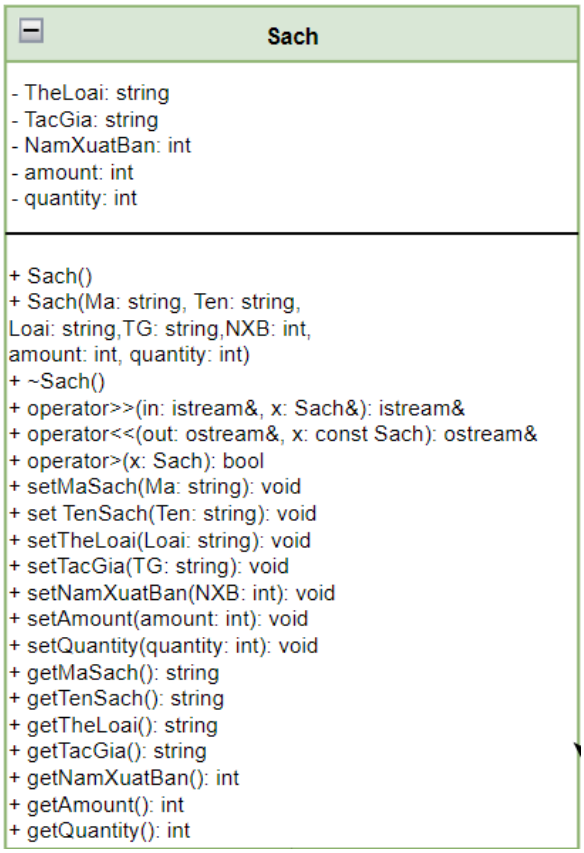
**2.2 Bản vẽ class diagram cho các lớp và cả chương trình.**



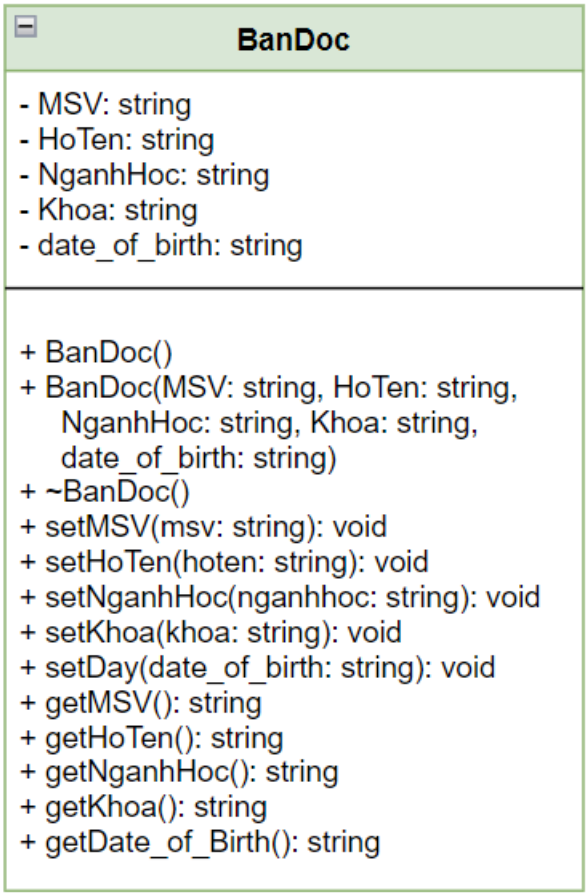

*Hình 1: Class Date*



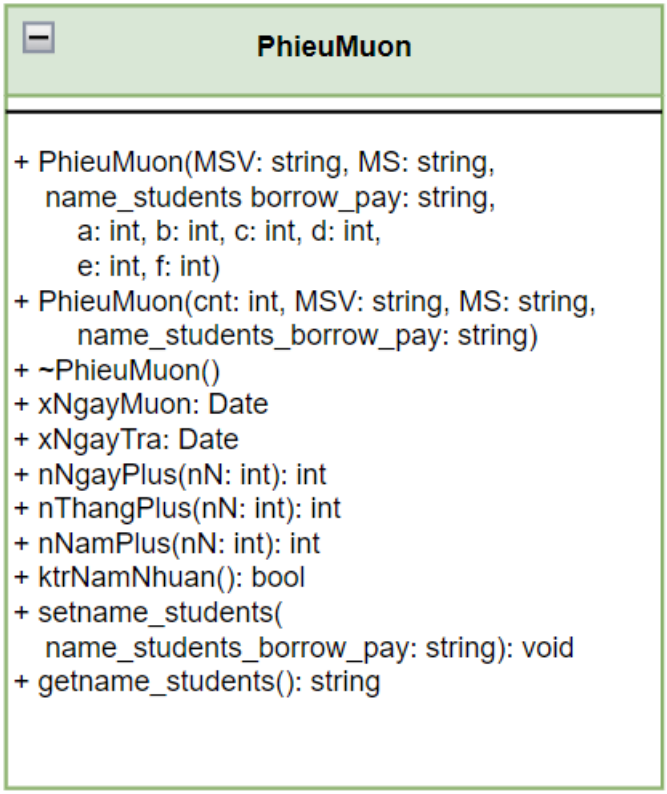

*Hình 2: Class xử lý chuỗi, kẻ bảng,..*



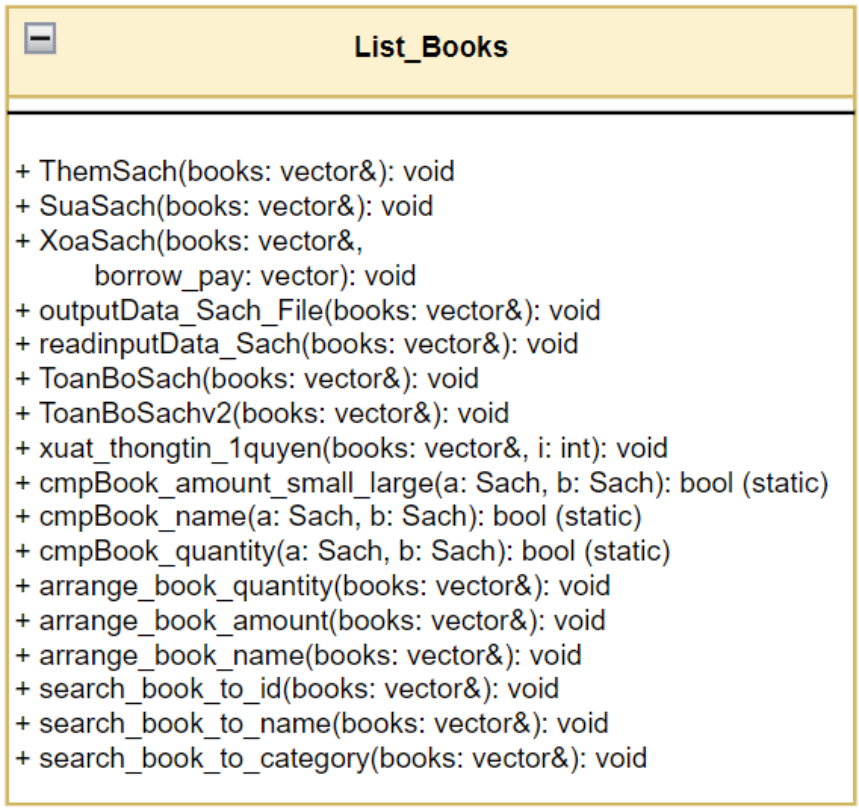

*Hình 3: Class Sach*



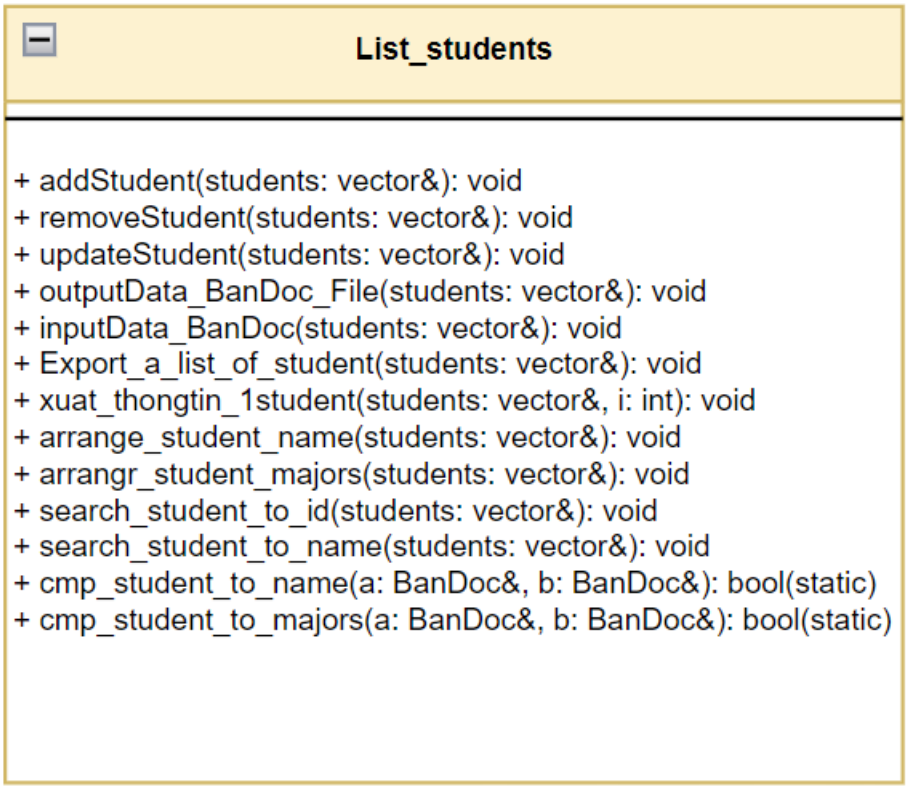

*Hình 4: Class BanDoc*



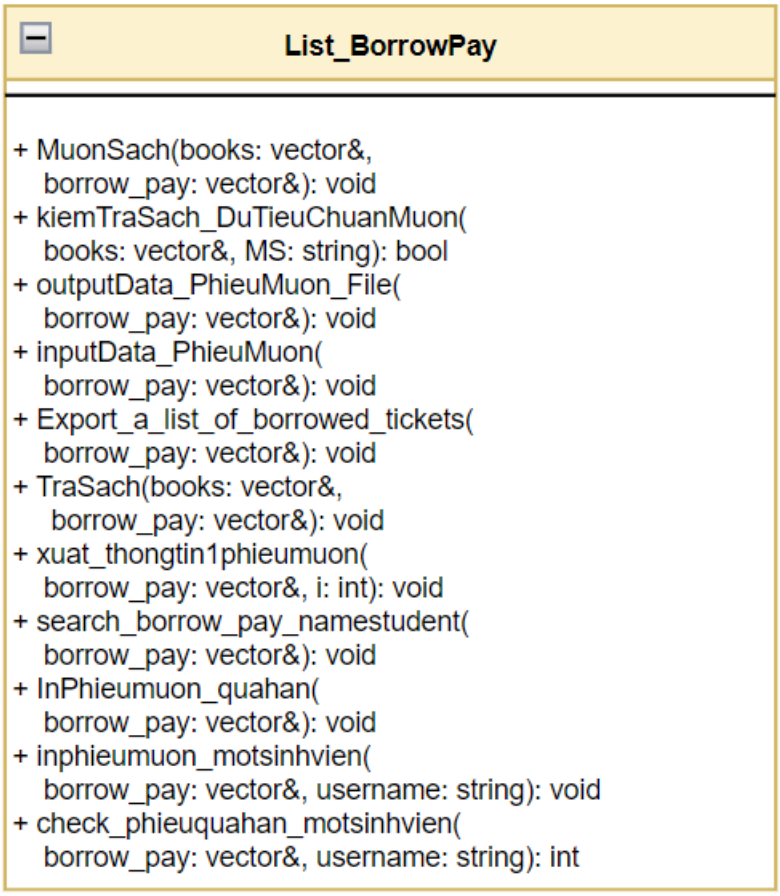

*Hình 5: Class PhieuMuon*



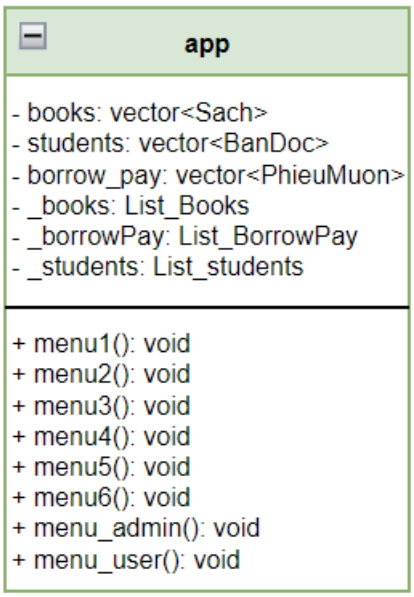

*Hình 6: Class ListBook*



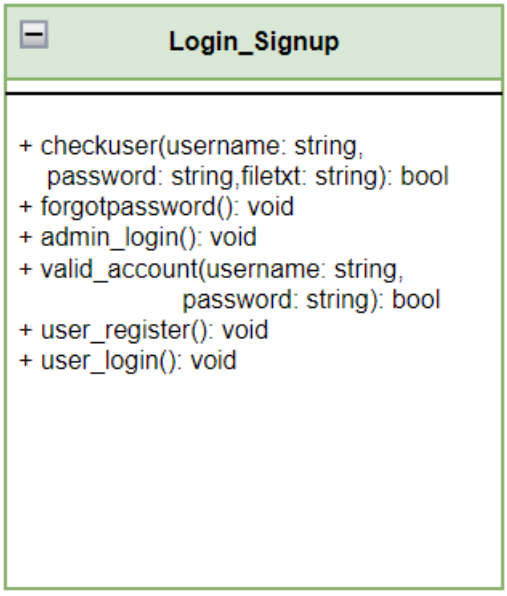

*Hình 7: Class List_students*




*Hình 8: Class ListBorrowPay*




*Hình 9: Class app*




*Hình 10: Class Login_Signup*

## III. Phân tích chức năng và độ phức tạp từng phương thức và cài đặt bằng C++ các Class.

Các phần chức năng, độ phức tạp và cài đặt code bằng C++ của từng phương thức trong mục II,III,IV em gộp thành 1 mục lớn.

## 3.1 Date

### 3.1.1 Chức năng:

– quản lý thông tin về ngày tháng năm

```
class Date
{
private:
    int day;
    int month;
    int year;

public:
    Date(int day = 0, int month = 0, int year = 0)
    {
        this->day = day;
        this->month = month;
        this->year = year;
    }
    void setNgay(int d)
    {
        this->day = d;
    }
    void setThang(int m)
    {
        this->month = m;
    }
    void setNam(int y)
    {
        this->year = y;
    }
    int getNgay()
    {
        return this->day;
    }
    int getThang()
    {
        return this->month;
    }
    int getNam()
    {
        return this->year;
```

```
        }
        void xuat()
        {
            cout << this->day << "/" << this->month << "/" << this->year;
        }
        ~Date(){};
    };
```

**3.1.2 Phương thức:**

các hàm set, get lần lượt là câu lệnh thay đổi và trả về => 1 phép toán =>O(1).

1. **Date(int day = 0, int month = 0, int year = 0):**
   - Chức năng:Thiết lập giá trị cho ngày, tháng, và năm của đối tượng.
   - Độ phức tạp: O(1)
2. **void setNgay(int d):**
   - Chức năng: Đặt giá trị ngày cho đối tượng Date.
   - Độ phức tạp: O(1)
3. **void setThang(int m):**
   - Chức năng: Đặt giá trị tháng cho đối tượng Date.
   - Độ phức tạp: O(1)
4. **void setNam(int y):**
   - Chức năng: Đặt giá trị năm cho đối tượng Date.
   - Độ phức tạp: O(1)
5. **int getNgay():**
   - Chức năng: Trả về giá trị ngày của đối tượng Date.
   - Độ phức tạp: O(1)
6. **int getThang():**
   - Chức năng: Trả về giá trị tháng của đối tượng Date.
   - Độ phức tạp: O(1)
7. **int getNam():**
   - Chức năng: Trả về giá trị năm của đối tượng Date.
   - Độ phức tạp: O(1)

8. **void xuat():**
   - Chức năng:In ra giá trị của day, month, và year theo định dạng "ngày/tháng/năm".
   - Độ phức tạp: O(1)
9. **Destructor ~Date():**
   - Chức năng: giải phóng tài nguyên của lớp Date.
   - Độ phức tạp: O(1)

## 3.2 StringManipulator

### 3.2.1 Chức năng:

- class có các chức năng thao tác với chuỗi như căn giữa một chuỗi để kẻ bảng, chuẩn hóa tên,chuẩn hoá ngày sinh và lấy thông tin ngày giờ theo thời gian thực

### 3.2.2 Phương thức:

1. **`center`:**

```
/*! Kẻ bảng, căn giữa chuỗi */
string center(const string s, const int width)
{
    stringstream ss, spaces;
    int padding = width - s.size(); // đếm phần thừa để căn
    for (int i = 0; i < padding / 2; ++i)
        spaces << " ";
    ss << spaces.str() << s << spaces.str(); // format with padding

    // nếu padding là số lẻ thì thêm 1 dấu cách vào sau cùng ss.
    if (padding > 0 && padding % 2 != 0)
        ss << " ";
    return ss.str();
}
```

- Chức năng: Căn giữa một chuỗi trong một chiều rộng cho trước bằng cách thêm dấu cách ở cả hai bên.
- Độ phức tạp: O(n), trong đó n = width - độ dài của chuỗi,for sẽ chạy n/2 lần.

2. **`size_number`:**

```cpp
int size_number(int s) // tìm số lượng chữ số của s
{
    int cnt = 0;
    while (s > 0)
    {
        s /= 10;
        ++cnt;
    }
    return cnt;
}
```

- Chức năng: Đếm số chữ số trong một số nguyên.
- Độ phức tạp: O(log n), trong đó n là giá trị của biến

3. **`centerv2`:**

```cpp
// nếu data là số
string centerv2(const int s, const int width)
{
    stringstream ss, spaces;
    int padding = width - size_number(s);
    for (int i = 0; i < padding / 2; ++i)
        spaces << " ";
    ss << spaces.str() << s << spaces.str();
    if (padding > 0 && padding % 2 != 0)
        ss << " ";
    return ss.str();
}
```

- Chức năng: Căn giữa một số nguyên trong một chiều rộng cho trước nhất định.
- Độ phức tạp: O(n) trong đó n = width – O(log n).

4. **`convert`:**

```
string convert(string s)
{
    s[0] = toupper(s[0]);
    for (int i = 1; i < s.size(); ++i)
    {
        s[i] = tolower(s[i]);
    }
    return s;
}
```

– Chức năng: Chuyển ký tự đầu tiên của xâu thành chữ hoa và các ký tự còn lại thành chữ thường phục vụ cho hàm standardization.

– Độ phức tạp: O(n), trong đó n là độ dài của chuỗi đầu vào.

5. **standardization:**

```
string standardization(string s1)
{
    stringstream ss(s1);//Khởi tạo stringstream từ xâu s1
    vector<string> res;
    string tmp;
    //Đọc lần lượt các phần của xâu. Các phần tách nhau bởi dấu cách
    while (ss >> tmp)
    {
        res.push_back(tmp);
    }
    string key = "";
    for (int i = 0; i < res.size() - 1; ++i)
    {
        key += convert(res[i]) + " ";
    }
    key += convert(res[res.size() - 1]);
    return key;
}
```

– Chức năng: Chuẩn hóa một chuỗi bằng cách viết hoa chữ cái đầu tiên của mỗi từ.vd: hOAng dinh NAm -> Hoang Dinh Nam

– Độ phức tạp: O(n + k*p + q), với n là kích thước của chuỗi đầu vào, k là số từ trong vector res, p là độ dài của từ dài nhất trong vector res và q là tổng số ký tự trong chuỗi kết quả.=> O(n) với n là tổng số ký tự trong chuỗi đầu vào s1.

6. **convertv2:**

```
string convertv2(string s)
{
    stringstream ss(s);
    string tmp;
    vector<string> res;
    while (ss >> tmp)
    {
        res.push_back(tmp);
    }
    tmp = "";
    tmp += res[res.size() - 1];
    for (int i = 0; i < res.size() - 1; ++i)
    {
        tmp += res[i];
    }
    return tmp;
}
```

– Chức năng: Chuyển đổi chuỗi về tên trước họ sau (vd: hoang dinh nam -> namhoangdinh) để phục vụ cho việc sắp xếp các bạn đọc theo thứ tự alphabet

– Độ phức tạp: O(n), trong đó n là độ dài của chuỗi đầu vào.

7. **convert_date:**

```
string convert_date(string s)
{
    if (s[2] != '/')
        s = "0" + s;
    if (s[5] != '/')
        s.insert(3, "0");
    return s;
}
```

– Chức năng: Chuyển đổi chuỗi ngày sang định dạng chuẩn (dd/mm/yyyy) vd: 5/7/2023 -> 05/07/2023

– Độ phức tạp: O(n), vì insert trong string có độ phức tạp như vậy.

8. **convert_datev2:**

```
string convert_datev2(int day, int month, int year)
{
    string s;
    s += to_string(day) + "/" + to_string(month) + "/" + to_string(year);
    return convert_date(s);
}
```

– Chức năng: Chuyển đổi các thành phần int ngày, tháng và năm riêng lẻ thành chuỗi dd/mm/yyyy.

– Độ phức tạp: O(n) vì insert trong convert_date.

9. **get_time:**

```
// lấy thời gian thực----------------------------
string get_time()
{
    time_t now = time(0);
    tm *ltm = localtime(&now);
    return to_string(ltm->tm_hour) + ":" + to_string(ltm->tm_min) + ":" + to_string(ltm->tm_sec);
}
```

– Chức năng: Lấy thời gian hiện tại ở định dạng "hh:mm:ss".

– Độ phức tạp: O(1), vì nó chỉ liên quan việc lấy thời gian hiện tại và nối chuỗi.

10. **get_day:**

```
string get_day()
{
    time_t now = time(0);
    tm *ltm = localtime(&now);
    return convert_date(to_string(ltm->tm_mday) + "/" + to_string(1 + ltm->tm_mon) +
                                                  "/" + to_string(1900 + ltm->tm_year));
}
```

– Chức năng: Lấy ngày hiện tại ở định dạng "dd/mm/yyyy".

– Độ phức tạp: O(n), vì insert trong hàm convert_date.

## 3.3 Sach

### 3.3.1 Chức năng:

– định nghĩa đối tượng sách cho thư viện

```
class Sach : public StringManipulator
{
```

```
        private:
            string MaSach;
            string TenSach;
            string TheLoai;
            string TacGia;
            int NamXuatBan;
            int amount;
            int quantity;

        public:
            Sach() {}
            Sach(string Ma = "", string Ten = "", string Loai = "", string TG = "", int NXB
= 0, int amount = 0, int quantity = 0)
            {
                MaSach = Ma;
                TenSach = Ten;
                TheLoai = Loai;
                TacGia = TG;
                NamXuatBan = NXB;
                this->amount = amount;
                this->quantity = quantity;
            }
            ~Sach(){};
     friend istream& operator >> (istream& in, Sach &x){

        cout << "- Nhap ma sach: ";
        in.ignore();
    ms:
        cout << "- Nhap ma sach: ";
        in >> x.MaSach;
        if (x.MaSach.size() == 0)
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto ms;
        }
    ten:
        cout << "- Nhap ten sach: ";
        getline(in, x.TenSach);
        if (x.TenSach == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto ten;
        }
    tl:
        cout << "- Nhap The Loai: ";
```

```
        getline(in, x.TheLoai);
        if (x.TheLoai == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto tl;
        }
    tg:
        cout << "- Nhap ten tac gia: ";
        getline(in, x.TacGia);
        if (x.TacGia == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto tg;
        }
        cout << "- Nhap nam xuat ban: ";
        in >> x.NamXuatBan;
        cout << "- Nhap gia cuon sach: ";
        in >> x.amount;
        cout << "- Nhap so luong: ";
        in >> x.quantity;
        return in;
    }
friend ostream& operator << (ostream& out, const Sach x)
    {
        out << string(5 + 25 + 15 * 2 + 12 + 10 + 9 + 3 * 6, '_') << "\n";
        out << x.MaSach << " | "
             << x.TenSach << " | "
             << x.TheLoai << " | "
             << x.TacGia << " | "
             << x.NamXuatBan << " | "
             << x.amount << " | "
             << x.quantity << "\n";
             return out;
    }
    bool operator > (Sach x){
        return this->TenSach > x.TenSach;
    }

            void setMaSach(string Ma)
            {
                MaSach = Ma;
            }

            void setTenSach(string Ten)
            {
```

```
        TenSach = Ten;
    }

    void setTheLoai(string Loai)
    {
        TheLoai = Loai;
    }

    void setTacGia(string TG)
    {
        TacGia = TG;
    }

    void setNamXuatBan(int NXB)
    {
        this->NamXuatBan = NXB;
    }
    void setAmount(int amount)
    {
        this->amount = amount;
    }
    void setQuantity(int quantity)
    {
        this->quantity = quantity;
    }
    string getMaSach()
    {
        return MaSach;
    }

    string getTenSach()
    {
        return standardization(TenSach);
    }

    string getTheLoai()
    {
        return standardization(TheLoai);
    }

    string getTacGia()
    {
        return TacGia;
    }
```

```
        int getNamXuatBan()
        {
            return NamXuatBan;
        }
        int getAmount()
        {
            return amount;
        }
        int getQuantity()
        {
            return quantity;
        }
};
```

### 3.3.2 Phương thức :

1. **Sach**(string Ma = "", string Ten = "", string Loai = "", string TG = "", int NXB = 0, int amount = 0, int quantity = 0):
   - Chức năng:Thiết lập giá trị cho các thuộc tính MaSach, TenSach, TheLoai, TacGia, NamXuatBan, amount, và quantity.
   - Độ phức tạp: O(1), vì chỉ thực hiện các phép gán và trả giá trị.
2. **void setMaSach(string Ma):**
   - Chức năng : Đặt giá trị cho thuộc tính MaSach.
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
3. **void setTenSach(string Ten):**
   - Chức năng : Đặt giá trị cho thuộc tính TenSach.
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
4. **void setTheLoai(string Loai):**
   - Chức năng : Đặt giá trị cho thuộc tính TheLoai.
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
5. **void setTacGia(string TG):**
   - Chức năng : Đặt giá trị cho thuộc tính TacGia.
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
6. **void setNamXuatBan(int NXB):**
   - Chức năng : Đặt giá trị cho thuộc tính NamXuatBan.

- Độ phức tạp: O(1), vì chỉ thực hiện phép gán.

7. **void setAmount(int amount):**
   - Chức năng : Đặt giá trị cho thuộc tính amount (giá của sách).
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
8. **void setQuantity(int quantity):**
   - Chức năng :: Đặt giá trị cho thuộc tính quantity (số lượng sách của 1 quyển).
   - Độ phức tạp: O(1), vì chỉ thực hiện phép gán.
9. **string getMaSach():**
   - Chức năng : Lấy giá trị của thuộc tính MaSach.
   - Độ phức tạp: O(1), vì chỉ thực hiện phép trả giá trị.
10. **string getTenSach():**
    - Chức năng : Lấy giá trị của thuộc tính TenSach sau khi đã được chuẩn hóa (thông qua hàm standardization()).
    - Độ phức tạp: O(n), vì hàm standardization có độ phức tạp như vậy.
11. **string getTheLoai():**
    - Chức năng : Lấy giá trị của thuộc tính TheLoai.
    - Độ phức tạp: O(n) vì hàm standardization có độ phức tạp như vậy.
12. **string getTacGia():**
    - Chức năng : Lấy giá trị của thuộc tính TacGia.
    - Độ phức tạp: O(1), vì chỉ thực hiện phép trả giá trị.
13. **int getNamXuatBan():**
    - Chức năng : Lấy giá trị của thuộc tính NamXuatBan.
    - Độ phức tạp: O(1), vì chỉ thực hiện phép trả giá trị.
14. **int getAmount():**
    - Trả về giá trị của thuộc tính amount.
    - Độ phức tạp: O(1), vì chỉ thực hiện phép trả giá trị.
15. **int getQuantity():**
    - Chức năng : Lấy giá trị của thuộc tính quantity.
    - Độ phức tạp: O(1), vì chỉ thực hiện phép trả giá trị.

16. **istream& operator >> (istream& in, Sach &x)**
    - Chức năng : Là hàm bạn của lớp có tác dụng xây dựng lại phương thức nhập bằng toán tử >> cho đối tượng `Sach`.
    - Độ phức tạp: O(1), vì chỉ thực hiện cac thao tác nhập cơ bản.

17. **ostream& operator << (ostream& out, const Sach x)**
    - Chức năng : Là hàm bạn của lớp có tác dụng xây dựng lại phương thức nhập bằng toán tử << cho đối tượng `Sach`.
    - Độ phức tạp: O(1), vì chỉ thực hiện các thao tác xuất cơ bản.

18. **operator > (Sach x)**
    - Chức năng : Là hàm so sánh > trả về tên sách theo thứ tự alphabet.
    - Độ phức tạp: O(1), vì chỉ thực hiện phép toán cơ bản.

## 3.4 BanDoc

### 3.4.1 Chức năng:

- định nghĩa các đối tượng sinh viên trong thư viện.

```
        class BanDoc : public StringManipulator
        {
        private:
            string MSV;
            string HoTen;
            string NganhHoc;
            string Khoa; // Khoá
            string date_of_birth;

        public:
            BanDoc() {}
            BanDoc(string MSV = "", string HoTen = "", string NganhHoc = "", string
Khoa = "", string date_of_birth = "")
            {
                this->MSV = MSV;
                this->HoTen = HoTen;
                this->NganhHoc = NganhHoc;
                this->Khoa = Khoa;
                this->date_of_birth = date_of_birth;
            }
            ~BanDoc() {}
            void setDay(string date_of_birth)
            {
                this->date_of_birth = date_of_birth;
```

```
        }
        void setKhoa(string Khoa)
        {
            this->Khoa = Khoa;
        }
        void setMSV(string msv)
        {
            MSV = msv;
        }

        void setHoTen(string hoten)
        {
            HoTen = hoten;
        }

        void setNganhHoc(string nganhhoc)
        {
            NganhHoc = nganhhoc;
        }

        string getMSV()
        {
            return MSV;
        }

        string getHoTen()
        {
            return standardization(HoTen);
        }

        string getNganhHoc()
        {
            return NganhHoc;
        }
        string getKhoa()
        {
            return Khoa;
        }
        string getDate_of_Birth()
        {
            return convert_date(date_of_birth);
        }
    };
```

### 3.4.2 Phương thức:

1. **BanDoc():**
   - Chức năng: Hàm khởi tạo mặc định không tham số.
   - Độ phức tạp: O(1).
2. **BanDoc**(string MSV, string HoTen, string NganhHoc, string Khoa, string date_of_birth):
   - Chức năng: Hàm khởi tạo với các tham số.
   - Độ phức tạp: O(1).
3. **~BanDoc():**
   - Chức năng: Hàm hủy, giải phóng bộ nhớ nếu cần.
   - Độ phức tạp: O(1).
4. **setMSV(string msv):**
   - Chức năng: Thiết lập mã số sinh viên.
   - Độ phức tạp: O(1).
5. **setHoTen(string hoten):**
   - Chức năng: Thiết lập họ tên sinh viên.
   - Độ phức tạp: O(1).
6. **setNganhHoc(string nganhhoc):**
   - Chức năng: Thiết lập ngành học của sinh viên.
   - Độ phức tạp: O(1).
7. **setKhoa(string Khoa):**
   - Chức năng: Thiết lập khoá học của sinh viên.
   - Độ phức tạp: O(1).
8. **setDay(string date_of_birth):**
   - Chức năng: Thiết lập ngày sinh của sinh viên.
   - Độ phức tạp: O(1).
9. **getMSV():**
   - Chức năng: Trả về mã số sinh viên.
   - Độ phức tạp: O(1).

**10. getNganhHoc():**

- Chức năng: Trả về ngành học của sinh viên.
- Độ phức tạp: O(1).

**11. getKhoa():**

- Chức năng: Trả về khoá học của sinh viên.
- Độ phức tạp: O(1).

**12. getDate_of_Birth():**

- Chức năng: Trả về ngày sinh đã được định dạng.
- Độ phức tạp: O(n), vì cú pháp insert của string mất O(n) trong hàm convert_date.

**13. getHoTen():**

- Chức năng: Trả về họ tên đã chuẩn hóa của sinh viên.
- Độ phức tạp: O(n ), vì hàm standardization có độ phức tạp như vậy.

## 3.5 PhieuMuon

### 3.5.1 Chức năng:

- định nghĩa cho 1 phiếu mượn.

### 3.5.2 Phương thức:

**1. `PhieuMuon`:**

```
PhieuMuon(string MSV = "", string MS = "", string name_students_borrow_pay = "", int a = 0, int b = 0,
int c = 0, int d = 0, int e = 0, int f = 0) : BanDoc(MSV, name_students_borrow_pay), Sach(MS), xNgayMuon(a, b, c), xNgayTra(d, e, f)
{}
```

- Chức năng: Khởi tạo đối tượng với các giá trị mặc định, constructor này dùng để đọc file PhieuMuon.txt,nếu dùng constructor bên dưới thì day month year trong file sẽ bị đổi thành thời gian thực.
- Độ phức tạp: O(1).

**2. PhieuMuon**(int cnt, string MSV, string MS, string name_students_borrow_pay)

```
PhieuMuon(int cnt = 0, string MSV = "", string MS = "", string name_students_borrow_pay = "") : BanDoc(MSV, name_students_borrow_pay), Sach(MS)
{
    time_t t = time(0);
    struct tm *Now = localtime(&t);
    xNgayMuon.setNgay(Now->tm_mday);
    xNgayMuon.setThang(Now->tm_mon + 1);
    xNgayMuon.setNam(Now->tm_year + 1900);

    xNgayTra.setNgay(nNgayPlus(7));
    xNgayTra.setThang(nThangPlus(7));
    xNgayTra.setNam(nNamPlus(7));
}
```

– Chức năng: Khởi tạo đối tượng với các giá trị mặc định hoặc được chỉ định.

– Độ phức tạp: O(1)

### 3. ~PhieuMuon:

```
~PhieuMuon(){};
```

– Chức năng: giải phóng bộ nhớ (nếu có) của lớp PhieuMuon.

– Độ phức tạp: O(1) - thời gian không đổi.

### 4. nNgayPlus(int nN):

```
int PhieuMuon::nNgayPlus(int nN)
{
    time_t t = time(0);
    struct tm *ta = localtime(&t);
    ta->tm_mon += 1;
    ta->tm_year += 1900;

    int nTongNgay = ta->tm_mday + nN;
    if (ta->tm_mon == 1 || ta->tm_mon == 3 || ta->tm_mon == 5 || ta->tm_mon
== 7 || ta->tm_mon == 8 || ta->tm_mon == 10 || ta->tm_mon == 12)
    {
        if (nTongNgay > 31)
        {
            nTongNgay -= 31;
        }
    }
    else if (ta->tm_mon == 2)
    {
        int maxDays = (ktrNamNhuan()) ? 29 : 28;
        if (nTongNgay > maxDays)
        {
            nTongNgay -= maxDays;
        }
    }
    else
    {
        if (nTongNgay > 30)
        {
            nTongNgay -= 30;
        }
    }
    return nTongNgay;
}
```

– Chức năng: Tính ngày sau khi cộng nN ngày vào ngày hiện tại,giải thích qua : 1 tháng có 31 ngày: gọi tổng ngày = lấy ngày hiện tại + nN số ngày nhập vào. đem so sánh với số 31.Nếu nhỏ hơn hoặc bằng 31 thì tháng vẫn giữ nguyên, tổng ngày = ngày

Nếu lớn hơn 31 thì lấy tổng ngày trừ cho 31 -> số tháng tăng lên 1, số ngày là kết quả của phép trừ.

– Độ phức tạp: O(1) - vì nó liên quan đến các phép toán số học cơ bản.

**5. nThangPlus(int nN):**

```
int PhieuMuon::nThangPlus(int nN)
{
    time_t t = time(0);
    struct tm *ta = localtime(&t);
    ta->tm_mon += 1;
    ta->tm_year += 1900;

    int nTongNgay = ta->tm_mday + nN;
    if (ta->tm_mon == 1 || ta->tm_mon == 3 || ta->tm_mon == 5 || ta->tm_mon
== 7 || ta->tm_mon == 8 || ta->tm_mon == 10)
    {
        if (nTongNgay > 31)
        {
            ta->tm_mon += 1;
        }
    }
    else if (ta->tm_mon == 12)
    {
        if (nTongNgay > 31)
        {
            ta->tm_mon = 1;
        }
    }
    else if (ta->tm_mon == 2)
    {
        int maxDays = (ktrNamNhuan()) ? 29 : 28;
        if (nTongNgay > maxDays)
        {
            ta->tm_mon += 1;
        }
    }
    else
    {
        if (nTongNgay > 30)
```

```
            {
                ta->tm_mon += 1;
            }
        }
    return ta->tm_mon;
}
```

– Chức năng: Tính tháng sau khi cộng nN ngày vào ngày hiện tại.

– Độ phức tạp: O(1) - vì nó liên quan đến các phép toán số học cơ bản.

**6. nNamPlus(int nN):**

```
int PhieuMuon::nNamPlus(int nN)
{
    time_t t = time(0);
    struct tm *ta = localtime(&t);
    ta->tm_mon = ta->tm_mon + 1;
    ta->tm_year = ta->tm_year + 1900;
    int nTongNgay = ta->tm_mday + nN;
    if (ta->tm_mon == 12 && nTongNgay > 31)
    {
        ta->tm_year += 1;
    }
    return ta->tm_year;
}
```

– Chức năng: Tính năm sau khi cộng nN ngày vào ngày hiện tại.

– Độ phức tạp: O(1) - vì nó liên quan đến các phép toán số học cơ bản.

**7. ktrNamNhuan():**

```
bool PhieuMuon::ktrNamNhuan()
{
    time_t t = time(0);
    struct tm *ta = localtime(&t);
    ta->tm_year = ta->tm_year + 1900;
    if (ta->tm_year % 400 == 0 || ta->tm_year % 4 == 0 && ta->tm_year % 100
!= 0)
        return true;
    else
        return false;
}
```

– Chức năng: Kiểm tra xem năm hiện tại có phải là năm nhuận hay không.

– Độ phức tạp: O(1) - vì nó bao gồm các kiểm tra có điều kiện đơn giản.

## 3.6 List_Books

### 3.6.1 Chức năng:

– quản lý danh sách các sách.

### 3.6.2 Phương thức:

1. **ThemSach(vector <Sach> &books):**

```
void ThemSach(vector<Sach> &books)
    {
        int chon, index = 0;
        cout << "\t\tNhap them mot cuon sach" << endl;
        cout << "1.Them vao dau." << endl;
        cout << "2.Them vao giua." << endl;
        cout << "3.Them vao cuoi" << endl;
        cout << "_____________________________" << endl;

        do
        {
            cout << "Ban chon chuc nang: ";
            cin >> chon;
            if (chon < 1 || chon > 3)
                cout << "\tKhong co chuc nang nay, moi ban nhap lai!" << endl;
        } while (chon < 1 || chon > 3);

        if (chon == 2)
        {
            do
            {
                cout << "Nhap vi tri ban muon them vao: ";
                cin >> index;
                if (index < 0 || index > books.size())
                cout << "\tKhong ton tai vi tri day, moi ban nhap lai!" << endl;
            } while (index < 0 || index > books.size());
        }
        cout << endl;
        string MS, Ten, Loai, TG;
        int NXB;
        int amount;
        int quantity;
        cin.ignore();
    ms:
```

```
        cout << "- Nhap ma sach: ";
        getline(cin, MS);
        if (MS.size() == 0)
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto ms;
        }
        else
        {
            for (int i = 0; i < books.size(); ++i)
            {

                if (MS == books[i].getMaSach())
                {
                    int cnt = books[i].getQuantity();
                    books[i].setQuantity(cnt + 1);
                    cout << "\t Da tang so luong san pham " << books[i].getTenSach()
<< "!\n";
                    return;
                }
            }
        }
    ten:
        cout << "- Nhap ten sach: ";
        getline(cin, Ten);
        if (Ten == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto ten;
        }
        else
        {
            for (int i = 0; i < books.size(); ++i)
            {
                if (books[i].getTenSach() == Ten)
                {
                    cout << "\nDa ton tai ten quyen sach nay trong thu vien!\n";
                    cout << "========================\n";
                    cout << "- Nhan Enter de nhap lai: ";
                    cin.ignore();
                    cin.get();
                    goto ten;
                }
            }
        }
```

```
    tl:
        cout << "- Nhap The Loai: ";
        getline(cin, Loai);
        if (Loai == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto tl;
        }
    tg:
        cout << "- Nhap ten tac gia: ";
        getline(cin, TG);
        if (TG == "")
        {
            cout << "\t\t\tKhong duoc de trong !\n";
            goto tg;
        }
        cout << "- Nhap nam xuat ban: ";
        cin >> NXB;
        cout << "- Nhap gia cuon sach: ";
        cin >> amount;
        cout << "- Nhap so luong cuon sach them vao: ";
        cin >> quantity;
        Sach New(MS, Ten, Loai, TG, NXB, amount, quantity);
        if (chon == 3)
            books.push_back(New);
        else if (chon == 2)
        {
            auto it = books.begin() + index;
            books.insert(it, New);
        }
        else if (chon == 1)
        {
            auto it = books.begin();
            books.insert(it, New);
        }
        outputData_Sach_File(books);
        cout << "\t\tThem sach thanh cong !\n";
    }
```

- Chức năng: Thêm một cuốn sách mới vào đầu, giữa hoặc cuối danh sách.
- Độ phức tạp: O(n) : trong đó n là số lượng sách trong vector
  - Nhận thông tin đầu vào của người dùng.O(1)

- Kiểm tra các sách hiện có: Việc này mất O(2n) thời gian vì hàm lặp qua vector sách.
- Thêm một cuốn sách mới vào: cuối vector => O(1), đầu và giữa vector =>O(n).
- Ghi vào một tệp: Việc này mất O(n) thời gian, vì hàm phải duyệt qua cả vector books để ghi file.

⇨ 1 + 2n + n + n => ước lượng O(n).

**2. SuaSach(vector <Sach> &books):**

```
void SuaSach(vector<Sach> &books)
    {
        // Tìm và sửa theo mã sách
        cout << "\t\tSua thong tin sach" << endl;
        cout << endl;
        string Find;
        string MS, Ten, Loai, TG;
        int NXB;
        int amount;
        cout << "Nhap ma sach can sua: ";
        bool check_find_sach = false;
        cin.ignore();
        getline(cin, Find);
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == Find)
            {
                check_find_sach = true;
                xuat_thongtin_1quyen(books, i);
                cout << "_____________________________________________________" << endl;
            ms:
                cout << "- Nhap ma sach moi: ";
                getline(cin, MS);
                if (MS == "")
                {
                    cout << "\t\t\tKhong duoc de trong !\n";
                    goto ms;
                }
                for (int i = 0; i < books.size(); ++i)
                {

                    if (MS == books[i].getMaSach())
                    {
                        int cnt = books[i].getQuantity();
```

```
                            books[i].setQuantity(cnt + 1);
                            cout << "\t Da tang so luong san pham " <<
books[i].getTenSach() << "!\n";
                            return;
                        }
                    }
                ten:
                    cout << "- Nhap ten sach moi: ";
                    getline(cin, Ten);
                    if (Ten == "")
                    {
                        cout << "\t\t\tKhong duoc de trong !\n";
                        goto ten;
                    }
                    else
                    {
                        for (int i = 0; i < books.size(); ++i)
                        {
                            if (books[i].getTenSach() == Ten)
                            {
                                cout << "\nDa ton tai ten quyen sach nay trong thu
vien!\n";
                                cout << "========================\n";
                                cout << "- Nhan Enter de nhap lai: ";
                                cin.ignore();
                                cin.get();
                                goto ten;
                            }
                        }
                    }
            tl:
                cout << "- Nhap The Loai: ";
                getline(cin, Loai);
                if (Loai == "")
                {
                    cout << "\t\t\tKhong duoc de trong !\n";
                    goto tl;
                }
                    tg:
                        cout << "- Nhap ten tac gia moi: ";
                        getline(cin, TG);
                        if (TG == "")
                        {
                            cout << "\t\t\tKhong duoc de trong !\n";
```

```
                    goto tg;
                }
                cout << "- Nhap nam xuat ban moi: ";
                cin >> NXB;
                cout << "- Nhap gia cuon sach moi: ";
                cin >> amount;
                Sach change(MS, standardization(Ten), standardization(Loai), TG,
NXB, amount, 1);
                books[i] = change;
                break;
            }
        }
        if (!check_find_sach)
            cout << "- Khong ton tai ma sach can sua !\n";
        else
            outputData_Sach_File(books);
    }
```

- Chức năng: Sửa đổi thông tin về một cuốn sách cụ thể.
- Độ phức tạp: O(n^2) :
  - Nhận thông tin đầu vào của người dùng.
  - Tìm kiếm sách để chỉnh sửa: Việc này mất O(n) thời gian vì hàm lặp qua vector sách.(vòng for)
  - Xuất thông tin của quyển sách (xuat_thongtin_1quyen) : O(n)
  - Thay đổi thông tin của cuốn sách(vòng for): O(n).
  - Ghi vào một tệp: Việc này mất O(n) thời gian, vì hàm phải duyệt qua cả vector books để ghi file.

⇨ 1 * n * (n + n + n)

### 3. XoaSach(vector <Sach> &books, vector <PhieuMuon> &borrow_pay):

```
void XoaSach(vector<Sach> &books, vector<PhieuMuon> &borrow_pay)
    {
        string Find;
        cout << "- Nhap ma sach can xoa: ";
        cin >> Find;

        bool found = false;

        for (int i = 0; i < books.size(); ++i)
```

```
        {
            if (books[i].getMaSach() == Find)
            {
                // Kiểm tra xem cuốn sách đang được mượn hay không

                if (books[i].getQuantity())
                {
                    books.erase(books.begin() + i);
                    cout << "Da xoa sach co ma: " << Find << endl;
                }
                else
                {
                    cout << "Khong the xoa cuon sach nay vi dang co sinh vien muon!"
<< endl;
                }

                found = true;
                break;
            }
        }

        if (!found)
        {
            cout << "Khong tim thay cuon sach co ma: " << Find << endl;
        }
        else
            outputData_Sach_File(books);
    }
```

- Chức năng: Xóa một cuốn sách khỏi thư viện.
- Độ phức tạp: O(n^2) :
  - Nhận thông tin đầu vào của người dùng.O(1)
  - Tìm kiếm sách để xóa: Việc này mất O(n) thời gian vì hàm lặp qua vector sách.
  - Kiểm tra xem sách có sẵn để xóa hay không: O(1).
  - Xóa sách bằng vector: O(n).
  - Ghi vào một tệp: Việc này mất O(n) thời gian, vì hàm phải duyệt qua cả vector books để ghi file.

⇨ 1 + n * (1 + n + n) = > O(n^2)

4. **outputData_Sach_File(vector <Sach> &sach):**

```
void outputData_Sach_File(vector<Sach> &books){
        // lấy data từ books ghi mới vào Sach.txt
        ofstream File;
        File.open("Sach.txt");
        int cnt = 0;
        for (int i = 0; i < books.size(); ++i)
        {
            File << books[i].getMaSach() << "," << books[i].getTenSach()
                 << "," << books[i].getTheLoai() << "," << books[i].getTacGia()
<< "," << books[i].getNamXuatBan() << ","
                  << books[i].getAmount() << "," << books[i].getQuantity();
            if (cnt < books.size() - 1)
            {
                File << endl;
                cnt++;
            }
        }
        File.close();
    }
```

- Chức năng: Ghi dữ liệu sách vào file (“Sach.txt”).
- Độ phức tạp: O(n) - tuyến tính, trong đó n là số lượng sách.

5. **readinputData_Sach(vector <Sach> &books):**

```
void readinputData_Sach(vector<Sach> &books){
        // lấy data từ file Sach.txt đẩy vào books
        books.clear();
        string Ma, Name_book, TL, TG;
        int NamXB, amount, quantity;
        ifstream File("Sach.txt");
        if (!File.is_open())
            return;
        while (!File.eof())
        {
            getline(File, Ma, ',');
            getline(File, Name_book, ',');
            getline(File, TL, ',');
            getline(File, TG, ',');
            File >> NamXB;
            File.ignore(1, ',');
            File >> amount;
            File.ignore(1, ',');
```

```
            File >> quantity;
            File.ignore(1, '\n');
            Sach s(Ma, standardization(Name_book), standardization(TL), TG,
NamXB, amount, quantity);
            books.push_back(s);
        }
        File.close();
    }
```

- Chức năng: Đọc dữ liệu từ một file ("Sach.txt") và đẩy vào vector books.
- Độ phức tạp: O(n) - tuyến tính, trong đó n là số lượng sách.

### 6. ToanBoSach(vector <Sach> &books):

```
void ToanBoSach(vector<Sach> &books)
    {
        cout << endl;
        cout << center("STT", 5) << " | "
             << center("Name", 25) << " | "
             << center("The loai", 15) << " | "
             << center("Tac Gia", 15) << " | "
             << center("Nam xuat ban", 12) << " | "
             << center("Amount", 10) << " | "
             << center("Quantity", 9) << "\n";

        cout << string(5 + 25 + 15 * 2 + 12 + 10 + 9 + 3 * 6, '_') << "\n";
        if (books.size() == 0)
        {
            cout << "\t\tHien khong co quyen sach nao trong thu vien!" << endl;
            return;
        }
        for (int i = 0; i < books.size(); ++i)
        {
            cout << center(books[i].getMaSach(), 5) << " | "
                 << center(books[i].getTenSach(), 25) << " | "
                 << center(books[i].getTheLoai(), 15) << " | "
                 << center(books[i].getTacGia(), 15) << " | "
                 << centerv2(books[i].getNamXuatBan(), 12) << " | "
                 << centerv2(books[i].getAmount(), 10) << " | "
                 << centerv2(books[i].getQuantity(), 8) << "\n";
            if (i < books.size() - 1)
                cout << string(5 + 25 + 15 * 2 + 12 + 10 + 9 + 3 * 6, '_') <<
"\n";
        }
    }
```

- Chức năng: Hiển thị thông tin về tất cả các cuốn sách.
- Độ phức tạp: O(n * m) - tuyến tính, trong đó n độ dài của vector books, m là độ dài của xâu được truyền vào để căn giữa hàm center (or độ dài của số truyền vào centerv2)..

7. **arrange_book_quantity(vector <Sach> &books):**

```
static bool cmpBook_quantity(Sach &a, Sach &b)
    {
        return a.getQuantity() < b.getQuantity();
    }
    void arrange_book_quantity(vector<Sach> books)
    {
        sort(books.begin(), books.end(), cmpBook_quantity);
        cout << "\nDone sap xep sach theo tang dan so luong!\n";
        ToanBoSach(books);
    }
```

- Chức năng: sắp xếp sách theo số lượng. Giải thích static giúp chúng trở thành các hàm tĩnh của class và do đó, chúng có thể được sử dụng như làm hàm so sánh thông thường cho std::sort.
- Độ phức tạp: O(n log n) - do thao tác sắp xếp dùng hàm sort.

8. **arange_book_amount(vector <Sach> &books):**

```
static bool cmpBook_amount_small_large(Sach &a, Sach &b)
    {
        return a.getAmount() < b.getAmount();
    }
    void arrange_book_amount(vector<Sach> books)
    {
        sort(books.begin(), books.end(), cmpBook_amount_small_large);
        cout << "\nDone sap xep sach theo tang dan gia tien!\n";
        ToanBoSach(books);
    }
```

- Chức năng: sắp xếp sách theo giá tiền.
- Độ phức tạp: O(n log n) - do thao tác sắp xếp dùng hàm sort.

9. **arange_book_name(vector <Sach> &books):**

```
static bool cmpBook_name(Sach &a, Sach &b)
```

```
    {
        return a.convert(a.getTenSach()) < b.convert(b.getTenSach());
    }
    void arrange_book_name(vector<Sach> books)
    {
        sort(books.begin(), books.end(), cmpBook_name);
        cout << "\nDone sap xep sach theo tang dan name!\n";
        ToanBoSach(books);
    }
```

- Chức năng: sắp xếp sách theo alphabet thứ tự tên + họ.
- Độ phức tạp: O(n log n) - do thao tác sắp xếp dùng hàm sort.

**10. search_book_to_id(vector <Sach> &books):**

```
void search_book_to_id(vector<Sach> books)
    {
        string id;
        cout << "- Nhap ma sach ban can tim: ";
        cin >> id;
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == id)
            {
                cout << "\t\t[!] Da tim thay " << books[i].getQuantity() << "
quyen sach ban can tim!\n\n";
                xuat_thongtin_1quyen(books, i);
                return;
            }
        }
        cout << "\n[!] Khong tim thay quyen sach ban can tim!\n";
    }
```

- Chức năng: Tìm kiếm sách theo ID của nó.
- Độ phức tạp: O(n * m) – trong đó n là độ dài của vector, m là độ dài của xâu truyền vào hàm center.

**11. search_book_to_name(vector <Sach> &books):**

```
        void search_book_to_name(vector<Sach> &books)
            {
                string name;
                cin.ignore();
                cout << "- Nhap ten sach ban can tim: ";
```

```
                getline(cin, name);
                for (int i = 0; i < books.size(); ++i)
                {
                    if (books[i].getTenSach() == standardization(name))
                    {
                        cout << "\t\t[!] Da tim thay " << books[i].getQuantity() <<
" quyen sach ban can tim!\n\n";
                        xuat_thongtin_1quyen(books, i);
                        return;
                    }
                }
                cout << "\n[!] Khong tim thay quyen sach ban can tim!\n";
            }
```

- Chức năng: Tìm kiếm một cuốn sách theo tên của nó.
- Độ phức tạp: O(n^2) - tìm kiếm tuyến tính O(n) * ( center O(n) + standardization O(n) ).

**12. search_book_to_category(vector <Sach> &books):**

```
void search_book_to_category(vector<Sach> &books)
    {
        string category;
        cin.ignore();
        bool check = false;
        cout << "- Nhap the loai ban muon doc: ";
        getline(cin, category);
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getTheLoai() == standardization(category))
            {
                if (!check)
                {
                    cout << "\t\t[!] Da tim thay quyen sach co the loai " <<
standardization(category) << endl
                         << endl;
                    cout << center("STT", 5) << " | "
                         << center("Name", 25) << " | "
                         << center("The loai", 15) << " | "
                         << center("Tac Gia", 15) << " | "
                         << center("Nam xuat ban", 12) << " | "
                         << center("Amount", 10) << " | "
                         << center("Quantity", 9) << "\n";
                    check = true;
```

```
                }
                cout << string(5 + 25 + 15 * 2 + 12 + 10 + 9 + 3 * 6, '_') << "\n";
                cout << center(books[i].getMaSach(), 5) << " | "
                     << center(books[i].getTenSach(), 25) << " | "
                     << center(books[i].getTheLoai(), 15) << " | "
                     << center(books[i].getTacGia(), 15) << " | "
                     << centerv2(books[i].getNamXuatBan(), 12) << " | "
                     << centerv2(books[i].getAmount(), 10) << " | "
                     << centerv2(books[i].getQuantity(), 8) << "\n";
            }
        }
        if (!check)
            cout << "\n[!] Rat xin loi, thu vien khong co the loai ban muon doc!\n";
    }
```

- Chức năng: Tìm kiếm sách theo thể loại.
- Độ phức tạp: O(n^2) - tìm kiếm tuyến tính O(n) * (center O(n) + standardization O(n)).

**13. xuat_thongtin_1quyen(vector <Sach> &books, int i):**

```
void xuat_thongtin_1quyen(vector<Sach> &books, int i)
    {
        cout << center("STT", 5) << " | "
             << center("Name", 25) << " | "
             << center("The loai", 15) << " | "
             << center("Tac Gia", 15) << " | "
             << center("Nam xuat ban", 12) << " | "
             << center("Amount", 10) << " | "
             << center("Quantity", 9) << "\n";

        cout << string(5 + 25 + 15 * 2 + 12 + 10 + 9 + 3 * 6, '_') << "\n";
        cout << center(books[i].getMaSach(), 5) << " | "
             << center(books[i].getTenSach(), 25) << " | "
             << center(books[i].getTheLoai(), 15) << " | "
             << center(books[i].getTacGia(), 15) << " | "
             << centerv2(books[i].getNamXuatBan(), 12) << " | "
             << centerv2(books[i].getAmount(), 10) << " | "
             << centerv2(books[i].getQuantity(), 8) << "\n";
    }
```

- Chức năng: Hiển thị thông tin về một cuốn sách cụ thể.
- Độ phức tạp: O(n), vì hàm center có độ phức tạp như vậy.

## 3.7 List_Students

### 3.7.1 Chức năng:

- quản lý danh sách bạn đọc ra vào thư viện.

### 3.7.2 Phương thức:

1. **addStudent(vector<BanDoc> &students):**

```
void addStudent(vector<BanDoc> &students)
    {
        cout << "\t\tNhap them mot ban doc" << endl;
        cout << endl;
        string MSV, Ten, NganhHoc;
        string Khoa;
        string date_of_birth;
        cin.ignore();
    msv:
        cout << "- Nhap ma sinh vien: ";
        getline(cin, MSV);
        if (MSV == "")
        {
            cout << "\t\t\tKhong duoc de trong. Nhap lai!\n";
            goto msv;
        }
        for (int i = 0; i < students.size(); ++i)
        {

            if (MSV == students[i].getMSV())
            {
                goto ten;
            }
        }
    ten:
        cout << "- Nhap ten sinh vien: ";
        getline(cin, Ten);
        if (Ten == "")
        {
            cout << "\t\t\tKhong duoc de trong. Nhap lai!\n";
            goto ten;
        }
    nh:
        cout << "- Nhap nganh hoc: ";
        getline(cin, NganhHoc);
        if (NganhHoc == "")
        {
```

```
            cout << "\t\t\tKhong duoc de trong. Nhap lai!\n";
            goto nh;
        }
    khoav2:
        cout << "- Nhap khoa hoc: ";
        cin >> Khoa;
        if (Khoa == "")
        {
            cout << "\t\t\tKhong duoc de trong. Nhap lai!\n";
            goto khoav2;
        }
        cout << "- Nhap ngay thang nam sinh theo dinh dang dd/mm/yyyy: ";
        cin >> date_of_birth;
        BanDoc New(MSV, standardization(Ten), NganhHoc, Khoa,
convert_date(date_of_birth));
        students.push_back(New);
        cout << "\t\tThem ban doc thanh cong !\n";
    }
```

– Chức năng:: Thêm dữ liệu một sinh viên

– Độ phức tạp: O(2n) => O(n) , với n là số lượng sinh viên trong danh sách. Do phương thức này có một vòng lặp để kiểm tra trùng mã sinh viên O(n) + hàm standardization O(n).

**2. removeStudent(vector<BanDoc> &students):**

```
    void removeStudent(vector<BanDoc> &students)
        {
            string MSV;
            cout << "- Nhap ma sinh vien can xoa: ";
            cin >> MSV;
            for (int i = 0; i < students.size(); ++i)
            {
                if (students[i].getMSV() == MSV)
                {
                    students.erase(students.begin() + i);
                    cout << "Da xoa sach co ma: " << MSV << endl;
                    return;
                }
            }
            cout << "\t\tKhong tim thay ma sinh vien: " << MSV << endl;
        }
```

– Chức năng:: Xoá dữ liệu một sinh viên theo mã sinh viên.

– Độ phức tạp: O(n^2), 1 vòng for O(n) * erase của vector O(n).

### 3. updateStudent(vector<BanDoc> &students):

```
void updateStudent(vector<BanDoc> &students)
    {
        // Tìm và sửa theo mã sách
        cout << "\t\tSua thong tin Ban doc" << endl;
        cout << endl;
        string MSV, Ten, NganhHoc, Khoa;
        string date_of_birth;
        cout << "Nhap ma sinh vien can sua: ";
        cin.ignore();
        cin >> MSV;
        cin.ignore();
        for (int i = 0; i < students.size(); ++i)
        {
            if (students[i].getMSV() == MSV)
            {
                xuat_thongtin_1student(students, i);
                cout << "_____________________________" << endl;
            ms:
                cout << "- Nhap ma sinh vien moi: ";
                getline(cin, MSV);
                if (MSV == "")
                {
                    cout << "\t\t\tKhong duoc de trong !\n";
                    goto ms;
                }
            ten:
                cout << "- Nhap ten sinh vien moi: ";
                getline(cin, Ten);
                if (Ten == "")
                {
                    cout << "\t\t\tKhong duoc de trong !\n";
                    goto ten;
                }
            nh:
                cout << "- Nhap nganh hoc moi: ";
                getline(cin, NganhHoc);
                if (NganhHoc == "")
                {
                    cout << "\t\t\tKhong duoc de trong !\n";
                    goto nh;
                }
            khoav2:
```

```
                cout << "- Nhap Khoa hoc moi: ";
                cin >> Khoa;
                if (Khoa == "")
                {
                    cout << "Khong duoc de trong. Nhap lai !\n";
                    goto khoav2;
                }
                cout << "- Nhap ngay sinh moi: ";
                cin >> date_of_birth;
                BanDoc change(MSV, standardization(Ten), NganhHoc, Khoa,
convert_date(date_of_birth));
                students[i] = change;
                return;
            }
        }
        cout << "- Khong ton tai ma sinh vien can sua !\n";
    }
```

- Chức năng:: sửa đổi thông tin của một sinh viên
- Độ phức tạp: O(n^2) vì ta ước lượng:
  - 1 vòng for duyệt tuyến tĩnh O(n).
    - Hàm xuat_thongtin_1student để xuất thông tin 1 sinh viên O(n)
    - Hàm standardization O(n)

⇨ n * (n + n) => O(n^2).

**4. Export_a_list_of_student(vector<BanDoc> &students):**

```
    void Export_a_list_of_student(vector<BanDoc> &students) {
        cout << endl;
        cout << center("STT", 5) << " | "
             << center("Ma sinh vien", 15) << " | "
             << center("Ho Ten", 25) << " | "
             << center("Nganh Hoc", 15) << " | "
             << center("Khoa", 10) << " | "
             << center("Ngay sinh", 12) << "\n";
        cout << string(5 + 15 * 2 + 25 + 10 + 12 + 5 * 3, '_') << "\n";
        if (students.size() == 0){
            cout << "\t\tHien khong co ban doc nao trong thu vien!\n";
            return;
        }
        for (int i = 0; i < students.size(); ++i){
            cout << centerv2(i + 1, 5) << " | "
                 << center(students[i].getMSV(), 15) << " | "
                 << center(students[i].getHoTen(), 25) << " | "
```

```
            << center(students[i].getNganhHoc(), 15) << " | "
            << center(students[i].getKhoa(), 10) << " | "
            << center(students[i].getDate_of_Birth(), 11) << "\n";
        if (i < students.size() - 1)
            cout << string(5 + 15 * 2 + 25 + 10 + 12 + 5 * 3, '_') << "\n";
    }
}
```

– Chức năng:: Xuất toàn bộ danh sách bạn đọc trong thư viện.

– Độ phức tạp: O(n * m), với n là số lượng sinh viên trong danh sách, m là độ dài của chuỗi truyền vào hàm center.

**5. xuat_thongtin_1student(vector<BanDoc> &students, int i):**

```
void xuat_thongtin_1student(vector<BanDoc> &students, int i){
    cout << center("STT", 5) << " | "
         << center("Ma sinh vien", 15) << " | "
         << center("Ho Ten", 25) << " | "
         << center("Nganh Hoc", 15) << " | "
         << center("Khoa", 10) << " | "
         << center("Ngay sinh", 15) << "\n";
    cout << string(5 + 15 * 3 + 25 + 10 + 5 * 3, '_') << "\n";
    cout << centerv2(i + 1, 5) << " | "
         << center(students[i].getMSV(), 15) << " | "
         << center(students[i].getHoTen(), 25) << " | "
         << center(students[i].getNganhHoc(), 15) << " | "
         << center(students[i].getKhoa(), 10) << " | "
         << center(students[i].getDate_of_Birth(), 15) << "\n";
}
```

– Chức năng:: Xuất toàn thông tin của 1 bạn đọc.

– Độ phức tạp: O(n ), với n là độ dài của chuỗi truyền vào hàm center.

**6. arrange_student_name(vector<BanDoc> &students):**

```
    static bool cmp_student_to_name(BanDoc &a, BanDoc &b)
    {
        return a.convertv2(a.getHoTen()) < b.convertv2(b.getHoTen());
    }
    void arrange_student_name(vector<BanDoc> students)
    {
        sort(students.begin(), students.end(), cmp_student_to_name);
        cout << "\nDone! Da sap xep name ban doc theo thu tu alphabet\n";
        Export_a_list_of_student(students);
```

```
        }
```

- Chức năng:: Sắp xếp bạn đọc theo thứ tự alphabet tên + họ
- Độ phức tạp: O(n log n), với n là số lượng sinh viên trong danh sách. Phương thức này sử dụng thuật toán sắp xếp (sort) có sẵn để sắp xếp danh sách sinh viên theo tên.

7. **`search_student_to_id` và `search_student_to_name`:**

```
void search_student_to_id(vector<BanDoc> &students) {
        string id;
        cout << "- Nhap ma sinh vien ban doc ban can tim: ";
        cin >> id;
        for (int i = 0; i < students.size(); ++i)
        {
            if (students[i].getMSV() == id)
            {
                cout << "\t\t\t[!] : Da tim thay ban doc co id " << id << "\n\n";
                xuat_thongtin_1student(students, i);
                return;
            }
            cout << "\n[!] : Khong tim thay ban doc co id = " << id << "\n";
        }
    }
    void search_student_to_name(vector<BanDoc> &students) {
        string name;
        cout << "- Nhap ten sinh vien ban doc ban can tim: ";
        cin.ignore();
        getline(cin, name);
        for (int i = 0; i < students.size(); ++i)
        {
            if (students[i].getHoTen() == standardization(name))
            {
                cout << "\t\t\t[!] : Da tim thay ban doc co name " << name <<
"\n\n";
                xuat_thongtin_1student(students, i);
                return;
            }
        }
        cout << "\n[!] : Khong ton tai ban doc co name " << name << " trong thu
vien\n";
    }
```

- Chức năng:: tìm kiếm thông tin sinh viên theo id và tên.

– Độ phức tạp: O(n), với n là số lượng sinh viên trong danh sách. Cả hai phương thức này duyệt tuyến tĩnh qua danh sách để tìm kiếm sinh viên theo mã sinh viên, tên sinh viên.

**8. arrange_student_majors(vector<BanDoc> &students):**

```
static bool cmp_student_to_majors(BanDoc &a, BanDoc &b)
    {
        return a.getNganhHoc() < b.getNganhHoc();
    }
    void arrange_student_majors(vector<BanDoc> students)
    {
        sort(students.begin(), students.end(), cmp_student_to_majors);
        cout << "\nDone! Da sap xep cac ban doc theo nganh\n";
        Export_a_list_of_student(students);
    }
```

– Chức năng:: Sắp xếp sinh viên theo thứ tự tăng dần ngành học.

– Độ phức tạp: O(n log n), với n là số lượng sinh viên trong danh sách, sử dụng thuật toán sắp xếp (sort) để sắp xếp danh sách sinh viên theo ngành.

## 3.8 List_BorrowPay

### 3.8.1 Chức năng:

– quản lý các phiếu mượn sách.

### 3.8.2 Phương thức:

**1. MuonSach(vector<Sach> &books, vector<PhieuMuon> &borrow_pay):**

```
void MuonSach(vector<Sach> &books, vector<PhieuMuon> &borrow_pay)
    {
        // Quản lý theo mã sinh viên và mã sách
        cout << "\t\t[!] : Nhap N de thoat.\n\n";
        string nMS = ""; // mã sách
        int check;
        while (true)
        {
            cout << "- Nhap ma sach: ";
            cin >> nMS;
            if (nMS == "")
            {
                cout << "Khong duoc de trong. Nhap lai!\n";
                continue;
            }
            else if (nMS == "N")
```

```
            {
                cout << "========================\n";
                cout << "Ban chon thoat !\n";
                return;
            }
            check = kiemTraSach_DuTieuChuanMuon(books, nMS);
            if (check == 1)
            {
                break;
            }
            else if (check == 2)
            {
                cout << "\t\tSach hien dang co ban doc khac muon.\n";
                continue;
            }
            else
            {
                cout << "\t\tMa sach khong ton tai!.\n";
                continue;
            }
        }
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == nMS)
            {
                cout << "\t\t" << nMS << " : " << books[i].getTenSach() << " |
quantity : " << books[i].getQuantity() << endl;
            }
        }
        string nMSV = "";
        while (true)
        {
            cout << "- Nhap ma ban doc can muon sach: ";
            cin >> nMSV;
            if (nMSV == "")
            {
                cout << "Khong duoc de trong ! \n";
                continue;
            }
            else if (nMSV == "N")
            {
                cout << "========================\n";
                cout << "Ban chon thoat !\n";
                return;
            }
```

```
        else
        {
            break;
        }
    }
    cin.ignore();
    string name_students_borrow_pay = "";
    while (true)
    {
        cout << "- Nhap ten ban doc muon sach: ";
        getline(cin, name_students_borrow_pay);
        if (name_students_borrow_pay == "")
        {
            cout << "Khong duoc de trong ! \n";
            continue;
        }
        else if (name_students_borrow_pay == "N")
        {
            cout << "========================\n";
            cout << "Ban chon thoat !\n";
            return;
        }
        else
        {
            break;
        }
    }
    PhieuMuon pm(0, nMSV, nMS, standardization(name_students_borrow_pay));
    borrow_pay.push_back(pm);
    cout << "_____________________________________\n";
    cout << "Yeu cau cua ban dang duoc thuc hien..." << endl;
    cout << "MSV: " << nMSV << endl;
    cout << "Ho ten: " << name_students_borrow_pay << endl;
    cout << "Ma sach: " << nMS << endl;
    cout << "Thoi gian: " << get_time() << endl;
    cout << "Ngay: " << get_day() << endl;
    cout << "Tao phieu muon thanh cong!\n";
    cout << "_____________________________________\n";
    ofstream File;
    File.open("PhieuMuon.txt", ios::app);
    int i = borrow_pay.size() - 1;
    if (i == 0)
    {
        File
```

```
                << borrow_pay[i].getMSV() << "," << borrow_pay[i].getMaSach() << ","
<< borrow_pay[i].getHoTen() << "," << borrow_pay[i].xNgayMuon.getNgay() << ","
                << borrow_pay[i].xNgayMuon.getThang() << "," <<
borrow_pay[i].xNgayMuon.getNam() << "," << borrow_pay[i].xNgayTra.getNgay() << ","
                << borrow_pay[i].xNgayTra.getThang() << "," <<
borrow_pay[i].xNgayTra.getNam();
        }
        else
            File << "\n"
                 << borrow_pay[i].getMSV() << "," << borrow_pay[i].getMaSach() << ","
<< borrow_pay[i].getHoTen() << "," << borrow_pay[i].xNgayMuon.getNgay() << ","
                 << borrow_pay[i].xNgayMuon.getThang() << "," <<
borrow_pay[i].xNgayMuon.getNam() << "," << borrow_pay[i].xNgayTra.getNgay() << ","
                 << borrow_pay[i].xNgayTra.getThang() << "," <<
borrow_pay[i].xNgayTra.getNam();
        File.close();
        // Sửa quantity của quyển sách vừa mượn
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == nMS)
            {
                int quantity = books[i].getQuantity();
                books[i].setQuantity(quantity - 1);
            }
        }
        outputData_Sach_File(books);
    }
```

- Chức năng: Thực hiện quá trình mượn sách, lưu thông tin phiếu mượn vào vector `borrow_pay` và cập nhật thông tin sách trong vector `books`.

- Độ phức tạp: O(n).

**2. kiemTraSach_DuTieuChuanMuon(vector<Sach> &books, string MS):**

```
int kiemTraSach_DuTieuChuanMuon(vector<Sach> &books, string MS)
    {
        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == MS)
            {
                if (books[i].getQuantity() > 0)
                    return 1;
                else
```

```
                    return 2;
                }
            }
            return 0;
        }
```

- Chức năng: Kiểm tra xem một quyển sách có đủ tiêu chuẩn để mượn hay không (còn số lượng).

- Độ phức tạp: O(n), với n là số lượng sách trong vector `books`.

### 3. outputData_PhieuMuon_File(vector<PhieuMuon> &borrow_pay):

```
void outputData_PhieuMuon_File(vector<PhieuMuon> &borrow_pay)
    {
        ofstream File;
        File.open("PhieuMuon.txt");
        int cnt = 0;
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            File
                << borrow_pay[borrow_pay.size() - 1].getMSV() << "," <<
borrow_pay[borrow_pay.size() - 1].getMaSach() << "," <<
borrow_pay[borrow_pay.size() - 1].getHoTen() << "," <<
borrow_pay[borrow_pay.size() - 1].xNgayMuon.getNgay() << ","
                << borrow_pay[borrow_pay.size() - 1].xNgayMuon.getThang() << ","
<< borrow_pay[borrow_pay.size() - 1].xNgayMuon.getNam() << "," <<
borrow_pay[borrow_pay.size() - 1].xNgayTra.getNgay() << "," <<
borrow_pay[borrow_pay.size() - 1].xNgayTra.getThang() << ","
                << borrow_pay[borrow_pay.size() - 1].xNgayTra.getNam();
            if (cnt < borrow_pay.size() - 1)
            {
                File << endl;
                cnt++;
            }
        }
        File.close();
    }
```

- Chức năng: Ghi thông tin phiếu mượn vào file "PhieuMuon.txt".

- Độ phức tạp: O(n), với n là số lượng phiếu mượn trong vector `borrow_pay`.

**4. inputData_PhieuMuon(vector<PhieuMuon> &borrow_pay):**

```
void inputData_PhieuMuon(vector<PhieuMuon> &borrow_pay){
        borrow_pay.clear();
        string MSV = "";
        string MS = "";
        string name_students_borrow_pay = "";
        int nNgayMuon = 0;
        int nThangMuon = 0;
        int nNamMuon = 0;
        int nNgayTra = 0;
        int nThangTra = 0;
        int nNamTra = 0;
        ifstream File("PhieuMuon.txt");
        if (!File.is_open())
            return;
        while (!File.eof())
        {
            getline(File, MSV, ',');
            getline(File, MS, ',');
            getline(File, name_students_borrow_pay, ',');
            File >> nNgayMuon;
            File.ignore(1, ',');
            File >> nThangMuon;
            File.ignore(1, ',');
            File >> nNamMuon;
            File.ignore(1, ',');
            File >> nNgayTra;
            File.ignore(1, ',');
            File >> nThangTra;
            File.ignore(1, ',');
            File >> nNamTra;
            File.ignore(1, '\n');
            PhieuMuon pm(MSV, MS, standardization(name_students_borrow_pay),
nNgayMuon, nThangMuon, nNamMuon, nNgayTra, nThangTra, nNamTra);
            borrow_pay.push_back(pm);
        }
        File.close();
    }
```

- Chức năng: Đọc thông tin phiếu mượn từ file "PhieuMuon.txt" và lưu vào vector `borrow_pay`.

- Độ phức tạp: O(n), với n là số lượng phiếu mượn trong file.

**5. Export_a_list_of_borrowed_tickets(vector<PhieuMuon> &borrow_pay):**

```
void Export_a_list_of_borrowed_tickets(vector<PhieuMuon> &borrow_pay){
        cout << "\t\t\t\tPhieu muon sach thu vien" << endl;
        cout << endl;
        cout << center("STT", 5) << " | "
             << center("Ho Ten", 25) << " | "
             << center("Ma sinh vien", 25) << " | "
             << center("Ma sach", 15) << " | "
             << center("Ngay muon", 14) << " | "
             << center("Ngay tra", 12) << "\n";
        cout << string(5 + 15 + 25 * 2 + 14 + 12 + 5 * 3, '_') << "\n";
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            cout << centerv2(i + 1, 5) << " | "
                 << center(borrow_pay[i].getHoTen(), 25) << " | "
                 << center(borrow_pay[i].getMSV(), 25) << " | "
                 << center(borrow_pay[i].getMaSach(), 15) << " | "
                 << center(convert_datev2(borrow_pay[i].xNgayMuon.getNgay(),
borrow_pay[i].xNgayMuon.getThang(), borrow_pay[i].xNgayMuon.getNam()), 14) << " |
"
                 << center(convert_datev2(borrow_pay[i].xNgayTra.getNgay(),
borrow_pay[i].xNgayTra.getThang(), borrow_pay[i].xNgayTra.getNam()), 12) << "\n";
            if (i < borrow_pay.size() - 1)
                cout << string(5 + 15 + 25 * 2 + 14 + 12 + 5 * 3, '_') << "\n";
        }
    }
```

- Chức năng: In ra danh sách các phiếu mượn trong thư viện.

- Độ phức tạp: O(n * m), với n là số lượng phiếu mượn trong vector borrow_pay, m là độ dài của chuỗi truyền vào hàm center(or độ dài của số truyền vào centerv2).

**6. TraSach(vector<Sach> &books, vector<PhieuMuon> &borrow_pay):**

```
void TraSach(vector<Sach> &books, vector<PhieuMuon> &borrow_pay){
        cin.ignore();
        cout << "\t\tTra sach" << endl;
        cout << "[!] : Nhap N de thoat.\n\n";
        string msv, ms, name;
        int check = -1;
    xmsv:
        cout << "- Nhap ma sinh vien: ";
```

```
        getline(cin, msv);
        if (msv == "N" || msv == "n")
        {
            cout << "\n==========================\n";
            cout << "\t\tBan chon thoat!\n";
            return;
        }
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            if (check == -1 && borrow_pay[i].getMSV() == msv)
            {
                check = i;
                cout << center("STT", 5) << " | "
                     << center("Ho Ten", 25) << " | "
                     << center("Ma sinh vien", 25) << " | "
                     << center("Ma sach", 15) << " | "
                     << center("Ngay muon", 14) << " | "
                     << center("Ngay tra", 12) << "\n";
            }
            if (borrow_pay[i].getMSV() == msv)
                xuat_thongtin1phieumuon(borrow_pay, i);
        }
        if (check != -1)
            goto xname;
        cout << "- Khong co ma sinh vien nao nhu vay da muon sach!\nNhap
lai:\n";
        goto xmsv;
    xname:
        cout << "- Nhap ten sinh vien: ";
        getline(cin, name);
        if (name == "N" || name == "n")
        {
            cout << "\n==========================\n";
            cout << "\t\tBan chon thoat!\n";
            return;
        }
        if (borrow_pay[check].getHoTen() == standardization(name))
        {
            goto xms;
        }
        cout << "- Nhap sai ten sinh vien muon sach!\nNhap lai:\n";
        goto xname;
    xms:
        cout << "- Nhap ma sach: ";
        cin >> ms;
```

```
        if (ms == "N" || ms == "n")
        {
            cout << "\n==========================\n";
            cout << "\t\tBan chon thoat!\n";
            return;
        }
        if (borrow_pay[check].getMaSach() == ms)
        {
            goto done;
        }
        cout << "- Nhap sai ma sach duoc muon!\nNhap lai:\n";
        goto xms;
    done:
        borrow_pay.erase(borrow_pay.begin() + check);

        for (int i = 0; i < books.size(); ++i)
        {
            if (books[i].getMaSach() == ms)
            {
                books[i].setQuantity(books[i].getQuantity() + 1);
            }
        }
        outputData_PhieuMuon_File(borrow_pay);
        outputData_Sach_File(books);
    }
```

- Chức năng: Thực hiện quá trình trả sách, cập nhật thông tin sách và xóa thông tin phiếu mượn.

- Độ phức tạp: O(n^2).

### 7. xuat_thongtin1phieumuon(vector<PhieuMuon> &borrow_pay, int i):

```
    void xuat_thongtin1phieumuon(vector<PhieuMuon> &borrow_pay, int i){
            cout << string(5 + 15 + 25 * 2 + 14 + 12 + 5 * 3, '_') << "\n";
            cout << centerv2(i + 1, 5) << " | "
                 << center(borrow_pay[i].getHoTen(), 25) << " | "
                 << center(borrow_pay[i].getMSV(), 25) << " | "
                 << center(borrow_pay[i].getMaSach(), 15) << " | "
                 << center(convert_datev2(borrow_pay[i].xNgayMuon.getNgay(),
borrow_pay[i].xNgayMuon.getThang(), borrow_pay[i].xNgayMuon.getNam()), 14) <<
" | "
```

```
                << center(convert_datev2(borrow_pay[i].xNgayTra.getNgay(),
borrow_pay[i].xNgayTra.getThang(), borrow_pay[i].xNgayTra.getNam()), 12) <<
"\n";
        }
```

- Chức năng: In thông tin của một phiếu mượn.

- Độ phức tạp: O(n).

### 8. search_borrow_pay_namestudent(vector<PhieuMuon> &borrow_pay):

```
void search_borrow_pay_namestudent(vector<PhieuMuon> &borrow_pay)
    {
        string name;
        cout << "- Nhap ten ban doc: ";
        cin.ignore();
        getline(cin, name);
        bool found = false;
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            if (found == false && borrow_pay[i].getHoTen() ==
standardization(name))
            {
                cout << "[!] : Da tim thay phieu muon cua ban doc " << name << "
!\n\n";
                cout << center("STT", 5) << " | "
                     << center("Ho Ten", 25) << " | "
                     << center("Ma sinh vien", 25) << " | "
                     << center("Ma sach", 15) << " | "
                     << center("Ngay muon", 14) << " | "
                     << center("Ngay tra", 12) << "\n";
                xuat_thongtin1phieumuon(borrow_pay, i);
                found = true;
            }
            else if (found && borrow_pay[i].getHoTen() == standardization(name))
            {
                xuat_thongtin1phieumuon(borrow_pay, i);
            }
        }
        if (!found)
            cout << "\n[!] : Khong ton tai phieu muon co ten ban doc " << name <<
" !\n\n";
    }
```

- Chức năng: Tìm kiếm và in ra thông tin phiếu mượn của một sinh viên dựa trên tên.

- Độ phức tạp: O(n^2).

9. **InPhieumuon_quahan(vector<PhieuMuon> &borrow_pay):**

```
void InPhieumuon_quahan(vector<PhieuMuon> &borrow_pay)
    {
        PhieuMuon current_day(0, "test", "test");
        int day = current_day.xNgayMuon.getNgay();
        int month = current_day.xNgayMuon.getThang();
        int year = current_day.xNgayMuon.getNam();
        cout << center("STT", 5) << " | "
             << center("Ho Ten", 25) << " | "
             << center("Ma sinh vien", 25) << " | "
             << center("Ma sach", 15) << " | "
             << center("Ngay muon", 14) << " | "
             << center("Ngay tra", 12) << "\n";
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            if (borrow_pay[i].xNgayTra.getNam() > year ||
(borrow_pay[i].xNgayTra.getNam() == year && (borrow_pay[i].xNgayTra.getThang() >
month ||

                    (borrow_pay[i].xNgayTra.getThang() == month &&

                     borrow_pay[i].xNgayTra.getNgay() > day))))
            {
                continue;
            }
            else
                xuat_thongtin1phieumuon(borrow_pay, i);
        }
    }
```

- Chức năng: In ra tất cả danh sách các phiếu mượn quá hạn bằng cách so sánh thời gian trả của phiếu với thời gian thực.

- Độ phức tạp: O(n^2).

10. **`inphieumuon_motsinhvien`:**

```
void inphieumuon_motsinhvien(vector<PhieuMuon> &borrow_pay, string username)
    {
        cout << center("STT", 5) << " | "
             << center("Ho Ten", 25) << " | "
             << center("Ma sinh vien", 25) << " | "
```

```
            << center("Ma sach", 15) << " | "
            << center("Ngay muon", 14) << " | "
            << center("Ngay tra", 12) << "\n";
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            if (borrow_pay[i].getMSV() == username)
            {
                xuat_thongtin1phieumuon(borrow_pay, i);
                return;
            }
        }
        cout << "\t\tBan chua muon quyen sach nao!\n";
    }
```

- Chức năng: In ra thông tin các phiếu mượn của một sinh viên dựa trên mã sinh viên.

- Độ phức tạp: O(n^2).

**11. `check_phieuquahan_motsinhvien`:**

```
int check_phieuquahan_motsinhvien(vector<PhieuMuon> &borrow_pay, string username){
        int cnt = 0;
        for (int i = 0; i < borrow_pay.size(); ++i)
        {
            if (borrow_pay[i].getMSV() == username)
            {
                ++cnt;
            }
        }
        return cnt;
    }
```

- Chức năng: Kiểm tra số lượng phiếu mượn quá hạn của một sinh viên dựa trên mã sinh viên.

- Độ phức tạp: O(n), với n là số lượng phiếu mượn trong vector `borrow_pay` của sinh viên đó.

## 3.9 Login_Signup

### 3.9.1 Chức năng:

– xử lý thông tin đăng nhập, đăng ký và chuyển hướng giữa các menu quản lý khác nhau).

### 3.9.2 Phương thức:

1. **Phương thức checkuser:**

```
bool checkuser(string username, string password, string filetxt)
    {
        string usernamev2, passwordv2;
        string fullname;
        ifstream File(filetxt);
        while (!File.eof())
        {
            getline(File, usernamev2, ' ');
            getline(File, passwordv2, ' ');
            getline(File, fullname, '\n');
            if (usernamev2 == username && passwordv2 == password)
            {
                return true;
            }
        }
        File.close();
        return false;
    }
```

- Chức năng: Kiểm tra tính hợp lệ của tên đăng nhập và mật khẩu so với dữ liệu trong file `database.txt` or `databasev2.txt`.
- Độ phức tạp: O(n), với n là số lượng tài khoản trong file text.

2. **Phương thức valid_account:**

```
bool valid_account(string username, string password)
    {
        string usernamev2, passwordv2, fullname;
        ifstream File("databasev2.txt");
        while (!File.eof())
        {
            getline(File, usernamev2, ' ');
            getline(File, passwordv2, ' ');
            getline(File, fullname, '\n');
            if (usernamev2 == username) // chỉ check tài khoản
            {
                return true;
            }
        }
        File.close();
        return false;
    }
```

- Chức năng: Kiểm tra tính hợp lệ của tài khoản dựa trên username, không kiểm tra mật khẩu.

– Độ phức tạp: O(n), với n là số lượng tài khoản trong file text.

### 3. Phương thức user_register:

```
void user_register() {
        string username, password, fullname;
    front:
        system("cls");
        cout << "\t\t"
             << "+----------------------+" << endl;
        cout << "\t\t"
             << "|     Register User    |" << endl;
        cout << "\t\t"
             << "+----------------------+" << endl;
    name:
        cout << "- Nhap username (Ma sinh vien): ";
        cin >> username;
        if (username == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto name;
        }
        else if (username.length() < 8)
        {
            cout << "\t\tUsername phai co 8 ky tu tro len. Nhap lai!\n";
            goto name;
        }
    pass:
        cout << "- Nhap password: ";
        cin >> password;
        if (password == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto pass;
        }
        else if (password.length() < 8)
        {
            cout << "\t\tPassword phai co 8 ky tu tro len. Nhap lai!\n";
            goto pass;
        }
        else
        {
            bool a[3] = {false, false, false};
            for (char c : password)
            {
                if (isdigit(c))
```

```
                {
                    a[0] = true;
                }
                if ('a' <= c && c <= 'z')
                {
                    a[1] = true;
                }
                if ('A' <= c && c <= 'Z')
                {
                    a[2] = true;
                }
                if (a[0] && a[1] && a[2])
                    break;
            }
            for (bool x : a)
            {
                if (x == false)
                {
                    cout << "\t\tPassword phai co day du cac ky tu a-z, A-Z, 0-9.
Nhap lai!\n";
                    goto pass;
                }
            }
        }
        if (!valid_account(username, password))
        {
            cin.ignore();
        fulln:
            cout << "- Nhap ten cua ban: ";
            getline(cin, fullname);
            if (fullname == "")
            {
                cout << "\t\tKhong duoc de trong. Nhap lai!\n";
                goto fulln;
            }
            else if (fullname.length() < 8)
            {
                cout << "\t\tFullname phai co 8 ky tu tro len. Nhap lai!\n";
                goto fulln;
            }
            fullname = standardization(fullname);
            ofstream File("databasev2.txt", ios::app);
            File << "\n"
                 << username << " " << password << " " << fullname;
            File.close();
```

```
            cout << endl
                 << "\t\tDang ky thanh cong!" << endl;
            cout << "=> Nhan Enter de den trang chu !";
            cin.ignore();
            cin.get();
            ok = true;
            menu_user(username, password, fullname);
            return;
        }
        else
        {
            int chon;
            cout << "\n\t\tTai khoan da ton tai!\n";
            cout << "_____________________________________" << endl;
            do
            {
                cout << "1. Dang ky.\n";
                cout << "2. Thoat!\n";
                cout << "Moi chon chuc nang: ";
                cin >> chon;
                switch (chon)
                {
                case 1:
                    goto back;
                case 2:
                    cout << "[2] : Thoat!\n";
                    cout << "\t\tBan chon thoat. Xin chao va hen gap lai!";
                    return;
                default:
                    cout << "\t\tNhap sai lua chon!\n";
                    break;
                }
            } while (chon != 2);
        }
    back:
        goto front;
    }
```

- Chức năng: Đăng ký tài khoản cho người dùng và chuyển đến menu user.
- Độ phức tạp: O(n), với n là số lượng tài khoản.

Phương thức này bao gồm các bước nhập tên đăng nhập và mật khẩu, kiểm tra tính hợp lệ (có thể có sự lồng ghép của các vòng lặp và điều kiện) xong thêm vào file text .

4. **Phương thức user_login:**

```
void user_login()
    {
        string username, password;
    lg:
        system("cls");
        cout << "\t\t"
             << "+--------------------+" << endl;
        cout << "\t\t"
             << "|     Login User     |" << endl;
        cout << "\t\t"
             << "+--------------------+" << endl;
        cout << "\t[!] : Nhap R de dang ky tai khoan" << endl;
    name:
        cout << "- Nhap username: ";
        cin >> username;
        if (username == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto name;
        }
        else if (username == "R" || username == "r")
        {
            user_register();
        }
        if (ok)
            return;
        else if (username.length() < 8)
        {
            cout << "\t\tUsername phai co 8 ky tu tro len. Nhap lai!\n";
            goto name;
        }
    pass:
        cout << "- Nhap password: ";
        cin >> password;
        if (password == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto pass;
        }
        else if (password == "R" || password == "r")
```

```
            user_register();
        if (ok)
            return;
        else if (password.length() < 8)
        {
            cout << "\t\tPassword phai co 8 ky tu tro len. Nhap lai!\n";
            goto pass;
        }
        else
        {
            bool a[3] = {false, false, false};
            for (char c : password)
            {
                if (isdigit(c))
                {
                    a[0] = true;
                }
                if ('a' <= c && c <= 'z')
                {
                    a[1] = true;
                }
                if ('A' <= c && c <= 'Z')
                {
                    a[2] = true;
                }
                if (a[0] && a[1] && a[2])
                    break;
            }
            for (bool x : a)
            {
                if (x == false)
                {
                    cout << "\t\tPassword phai co day du cac ky tu a-z, A-Z, 0-9.
Nhap lai!\n";
                    goto pass;
                }
            }
        }
        // nhập đúng định dạng xong thì đến bước check data
        bool check = false;
        string usernamev2, passwordv2;
        string fullname;
        ifstream File("databasev2.txt");
        while (!File.eof())
        {
```

```
            getline(File, usernamev2, ' ');
            getline(File, passwordv2, ' ');
            getline(File, fullname, '\n');
            if (usernamev2 == username && passwordv2 == password)
            {
                check = true;
                break;
            }
        }
        File.close();
        if (check)
        {
            cout << "\n\t\tDang nhap thanh cong!\n";
            cout << "=> Nhan Enter de den trang chu !";
            cin.ignore();
            cin.get();
            ok2 = true;
            menu_user(username, password, standardization(fullname));
        }
        else
        {
            cout << "\n_____________________________________________";
            cout << "\n\t\tTai khoan hoac mat khau khong dung!\n\n";
            int chon;
            do
            {
                cout << "1. Nhap lai\n";
                cout << "2. Thoat!\n";
                cout << "3. Dang ky\n";
                cout << "Moi chon chuc nang: ";
                cin >> chon;
                switch (chon)
                {
                case 1:
                    cout << "[1] : Nhap lai\n";
                    cout << "\t\tBan chon nhap lai. Quay lai trang login!";
                    goto lg;
                    break;
                case 2:
                    cout << "[2] : Thoat!\n";
                    cout << "\t\tBan chon thoat! Xin chao va hen gap lai !\n";
                    break;
                case 3:
                    cout << "[3] : Dang ky tai khoan\n";
                    user_register();
```

```
                break;
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            return;
        } while (chon != 2);
    }
}
```

- Chức năng: Đăng nhập cho người dùng và chuyển đến menu người dùng.
- Độ phức tạp: O(n) với n là số lượng tài khoản.

  Phương thức này gồm nhiều bước như nhập username, password, kiểm tra tính hợp lệ, và chuyển đến menu người dùng (có thể có sự lồng ghép của các vòng lặp và điều kiện).

**5. Phương thức admin_login:**

```
void admin_login() {
        string username, password;
    lg:
        system("cls");
        cout << "\t\t"
             << "+---------------------+" << endl;
        cout << "\t\t"
             << "|     Login Admin     |" << endl;
        cout << "\t\t"
             << "+---------------------+" << endl;
        cout << "\n[!] : Nhap T de vao web voi tu cach user!\n";
    name:
        cout << "- Nhap username: ";
        cin >> username;
        if (username == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto name;
        }
        else if (username == "T" || username == "t")
        {
            int chon;
            do
            {
                system("cls");
                cout << "\t\t"
```

```
                    << "+---------------+" << endl;
                cout << "\t\t"
                    << "|     User      |" << endl;
                cout << "\t\t"
                    << "+---------------+" << endl;
                cout << endl;
                cout << "1. LOGIN\n";
                cout << "2. SIGNUP\n";
                cout << "Moi chon chuc nang: ";
                cin >> chon;
                if (chon == 1)
                {
                    user_login();
                    return;
                }
                else if (chon == 2)
                {
                    user_register();
                    return;
                }
                else
                {
                    cout << "Nhap sai lua chon!\n";
                    cout << "=> Nhan Enter de tro ve !";
                    cin.ignore();
                    cin.get();
                }
            } while (chon != 1 && chon != 2);
        }
        if (ok || ok2) // nếu đã vào menu rồi mà quay về đây thì chắc chắn là bạn đã chọn chức năng thoát
            return;
        else if (username.length() < 8)
        {
            cout << "\t\tUsername phai co 8 ky tu tro len. Nhap lai!\n";
            goto name;
        }

    pass:
        cout << "- Nhap password: ";
        cin >> password;
        if (password == "")
        {
            cout << "\t\tKhong duoc de trong. Nhap lai!\n";
            goto pass;
```

```
            }
            else if (username == "T" || username == "t")
            {
                int chon;
                do
                {
                    system("cls");
                    cout << "\t\t"
                         << "+---------------+" << endl;
                    cout << "\t\t"
                         << "|     User      |" << endl;
                    cout << "\t\t"
                         << "+---------------+" << endl;
                    cout << endl;
                    cout << "1. LOGIN\n";
                    cout << "2. SIGNUP\n";
                    cout << "Moi chon chuc nang: ";
                    cin >> chon;
                    if (chon == 1)
                        user_login();
                    else if (chon == 2)
                        user_register();
                    else
                    {
                        cout << "Nhap sai lua chon!\n";
                        cout << "=> Nhan Enter de tro ve !";
                        cin.ignore();
                        cin.get();
                    }
                } while (chon != 1 && chon != 2);
            }
            if (ok || ok2) // nếu đã vào menu rồi mà quay về đây thì chắc chắn là
bạn đã chọn chức năng thoát
                return;
            else if (password.length() < 8)
            {
                cout << "\t\tPassword phai co 8 ky tu tro len. Nhap lai!\n";
                goto pass;
            }

            if (checkuser(username, password, "database.txt"))
            {
                cout << "\n\t\tDang nhap thanh cong!";
                flag = true;
                menu_admin();
```

```
            }
            else
            {
                cout << "\n\t\tTai khoan hoac mat khau khong dung!\n";
                cout << "_____________________________________________\n";
                int chon;
                do
                {
                    cout << "1. Nhap lai\n";
                    cout << "2. Thoat!\n";
                    cout << "3. Vao web voi tu cach user\n";
                    cout << "Moi chon chuc nang: ";
                    cin >> chon;
                    switch (chon)
                    {
                    case 1:
                        cout << "[1] : Nhap lai\n";
                        cout << "\t\tBan chon nhap lai. Quay lai trang login!";
                        goto lg;
                        break;
                    case 2:
                        cout << "[2] : Thoat!\n";
                        cout << "\t\tBan chon thoat! Xin chao va hen gap lai
!\n";
                        break;
                    case 3:
                        cout << "[3] : Vao web voi tu cach user!\n";
                        ok = true;
                        user_login();
                        break;
                    default:
                        cout << "\t\tNhap sai lua chon!\n";
                        cout << "=> Nhan Enter de tro ve !";
                        cin.ignore();
                        cin.get();
                        break;
                    }
                    if (ok)
                        return;
                } while (chon != 2);
            }
        }
```

– Chức năng: Đăng nhập và chuyển đến menu quản lý của admin.

– Độ phức tạp: O(n) với n là số lượng tài khoản.

– Phương thức này gồm nhiều bước như nhập username, password, kiểm tra tính hợp lệ, và chuyển đến menu quản lý (có thể có sự lồng ghép của các vòng lặp và điều kiện).

## 3.10 app.

### 3.10.1 Chức năng:

– lớp ứng dụng quản lý các menu và thực hiện các thao tác ở các lớp danh sách quản lý sách, bạn đọc, phiếu mượn để tổ chức mã nguồn.

### 3.10.2 Phương thức:

**1. menu1():**

```
void menu1() {
        int chon;
        do
        {
            system("cls");
            cout << "======================= MENU 1 ==========================\n";
            cout << "==                                                     ==\n";
            cout << "==        1.Them mot cuon sach.                        ==\n";
            cout << "==        2.Sua thong tin sach.                        ==\n";
            cout << "==        3.Xoa mot cuon sach.                         ==\n";
            cout << "==        4.Liet ke toan bo sach trong thu vien.       ==\n";
            cout << "==        5.Exit.                                      ==\n";
            cout << "=========================================================\n";
            cout << "=> Moi chon chuc nang: ";
            cin >> chon;
            cout << "=========================" << endl;
            switch (chon)
            {
            case 1:
                cout << "[1] : Them mot quyen sach\n";
                _books.ThemSach(books);
                break;
            case 2:
                cout << "[2] : Sua thong tin sach\n";
                _books.SuaSach(books);
                break;
            case 3:
                cout << "[3] : Xoa mot quyen sach\n";
                _books.XoaSach(books, borrow_pay);
                break;
```

```
            case 4:
                cout << "[4] : Liet ke toan bo sach trong thu vien\n";
                cout << "\t\t\t\tLiet ke toan bo sach trong thu vien" << endl;
                _books.ToanBoSach(books);
                break;
            case 5:
                cout << "[5] : Exit\n";
                return;
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            if (chon != 5)
            {
                cout << endl;
                cout << "========================" << endl;
                cout << "=> Nhan Enter de tro ve !";
                cin.ignore();
                cin.get();
            }
        } while (chon != 5);
    }
```

- Chức năng: Hiển thị menu quản lý sách và thực hiện các chức năng như thêm sách, sửa thông tin sách, xóa sách, liệt kê toàn bộ sách.
- Độ phức tạp: O(n), với n là số lượng sách hoặc số lượng phiếu mượn, tùy thuộc vào chức năng cụ thể.

**2. menu2():**

```
void menu2() {
        int chon;
        do
        {
            system("cls");
            cout << "======================= MENU 2 ==========================\n";
            cout << "==                                                     ==\n";
            cout << "==        1.Muon sach                                  ==\n";
            cout << "==        2.Tra sach                                   ==\n";
            cout << "==        3.Xuat danh sach phieu muon                  ==\n";
            cout << "==        4.Xuat danh sach phieu muon qua han          ==\n";
            cout << "==        5.Exit.                                      ==\n";
            cout << "=========================================================\n";
            cout << "=> Moi chon chuc nang: ";
```

```
            cin >> chon;
            cout << "========================" << endl;
            switch (chon)
            {
            case 1:
                cout << "[1] : Muon sach\n";
                _borrowPay.MuonSach(books, borrow_pay);
                break;
            case 2:
                cout << "[5] : Tra sach\n";
                _borrowPay.TraSach(books, borrow_pay);
                break;
            case 3:
                cout << "[3] : Xuat danh sach phieu muon\n";
                _borrowPay.Export_a_list_of_borrowed_tickets(borrow_pay);
                break;
            case 4:
                cout << "[4] : Xuat danh sach phieu muon qua han!\n";
                _borrowPay.InPhieumuon_quahan(borrow_pay);
                break;
            case 5:
                cout << "[5] : Exit\n";
                return;
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            if (chon != 5)
            {
                cout << endl;
                cout << "========================" << endl;
                cout << "=> Nhan Enter de tro ve !";
                cin.ignore();
                cin.get();
            }
        } while (chon != 5);
    }
```

- Chức năng: Hiển thị menu quản lý phiếu mượn sách và thực hiện các chức năng như mượn sách, trả sách, xuất danh sách phiếu mượn, xuất danh sách phiếu mượn quá hạn.
- Độ phức tạp: O(n), với n là số lượng sách hoặc số lượng phiếu mượn, tùy thuộc vào chức năng cụ thể.

### 3. menu3():

```
void menu3(){
        int chon;
        do
        {
            system("cls");
            cout << "======================= MENU 3 ==============================\n";
            cout << "==                                                         ==\n";
            cout << "==        1.Them mot ban doc vao thu vien.                 ==\n";
            cout << "==        2.Sua thong tin ban doc.                         ==\n";
            cout << "==        3.Xoa mot ban doc.                               ==\n";
            cout << "==        4.Liet ke toan bo sinh vien o trong thu vien.    ==\n";
            cout << "==        5.Exit.                                          ==\n";
            cout << "=============================================================\n";
            cout << "=> Moi chon chuc nang: ";
            cin >> chon;
            cout << "=========================" << endl;
            switch (chon)
            {
            case 1:
                cout << "[1] : Them mot ban doc\n";
                _students.addStudent(students);
                break;
            case 2:
                cout << "[2] : Sua thong tin ban doc\n";
                _students.updateStudent(students);
                break;
            case 3:
                cout << "[3] : Xoa mot ban doc\n";
                _students.removeStudent(students);
                break;
            case 4:
                cout << "[4] : Liet ke toan bo sinh vien o trong thu vien\n";
                cout << "\t\t\t\tIn ra cac ban doc trong thu vien hien tai" << endl;
                _students.Export_a_list_of_student(students);
                break;
            case 5:
                cout << "[5] : Exit\n";
                return;
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            if (chon != 5)
            {
```

```
            cout << endl;
            cout << "=========================" << endl;
            cout << "=> Nhan Enter de tro ve !";
            cin.ignore();
            cin.get();
        }
    } while (chon != 5);
}
```

- Chức năng: Hiển thị menu quản lý sinh viên ra vào thư viện và thực hiện các chức năng như thêm sinh viên, sửa thông tin sinh viên, xóa sinh viên, liệt kê toàn bộ sinh viên.
- Độ phức tạp: O(n), với n là số lượng sinh viên hoặc số lượng sách mượn, tùy thuộc vào chức năng cụ thể.

**4. menu4():**

```
void menu4() {
    int chon;
    do
    {
        system("cls");
        cout << "========================== MENU 4 ============================\n";
        cout << "==                                                          ==\n";
        cout << "==        1.Sap xep sach theo gia tien tang dan.            ==\n";
        cout << "==        2.Sap xep sach theo name sach tang dan.           ==\n";
        cout << "==        3.Sap xep danh sach cac ban doc theo nganh.       ==\n";
        cout << "==        4.Sap xep cac ban doc theo thu tu alphabet.       ==\n";
        cout << "==        5.Exit.                                           ==\n";
        cout << "==============================================================\n";
        cout << "=> Moi chon chuc nang: ";
        cin >> chon;
        cout << "=========================" << endl;
        switch (chon)
        {
        case 1:
            cout << "[1] : Sap xep sach theo thu tu tang dan gia tien\n";
            _books.arrange_book_amount(books);
            break;
        case 2:
            cout << "[2] : Sap xep sach theo thu tu tang dan ten sach\n";
            _books.arrange_book_name(books);
            break;
        case 3:
```

```
                cout << "[3] : Sap xep cac ban doc theo nganh\n";
                _students.arrange_student_majors(students);
                break;
            case 4:
                cout << "[4] : Sap xep cac ban doc theo thu tu alphabet tang dan theo ten +
ho\n";
                _students.arrange_student_name(students);
                break;
            case 5:
                cout << "[5] : Exit\n";
                return;
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            if (chon != 5)
            {
                cout << endl;
                cout << "=========================" << endl;
                cout << "=> Nhan Enter de tro ve !";
                cin.ignore();
                cin.get();
            }
        } while (chon != 5);
    }
```

- Chức năng: Hiển thị menu sắp xếp và thực hiện các chức năng sắp xếp sách theo giá tăng dần, sách theo tên, sinh viên theo tên.
- Độ phức tạp: O(n log n), với n là số lượng sách hoặc số lượng sinh viên, tùy thuộc vào chức năng cụ thể.

**5. menu5():**

```
void menu5() {
        int chon;
        do {
            system("cls");
            cout << "======================= MENU 5 ==========================\n";
            cout << "==                                                     ==\n";
            cout << "==        1.Tim kiem sach theo ma sach                 ==\n";
            cout << "==        2.Tim kiem sach theo name                    ==\n";
            cout << "==        3.Tim kiem theo the loai sach                ==\n";
            cout << "==        4.Tim kiem cac ban doc theo ma sinh vien     ==\n";
            cout << "==        5.Tim kiem ban doc theo name                 ==\n";
```

```
        cout << "==        6.Tim kiem phieu muon theo ten sinh vien.    ==\n";
        cout << "==        7.Exit.                                      ==\n";
        cout << "=============================================================\n";
        cout << "=> Moi chon chuc nang: ";
        cin >> chon;
        cout << "=========================" << endl;
        switch (chon)
        {
        case 1:
            cout << "[1] : Tim kiem sach theo ma sach\n";
            // switch case nó không cho nhập ở đây
            _books.search_book_to_id(books);
            break;
        case 2:
            cout << "[2] : Tim kiem sach theo ma name\n";
            _books.search_book_to_name(books);
            break;
        case 3:
            cout << "[3] : Tim kiem sach theo the loai\n";
            _books.search_book_to_category(books);
            break;
        case 4:
            cout << "[4] : Tim kiem ban doc theo ma sinh vien\n";
            _students.search_student_to_id(students);
            break;
        case 5:
            cout << "[5] : Tim kiem ban doc theo name\n";
            _students.search_student_to_name(students);
            break;
        case 6:
            cout << "[6] : Tim kiem ten sinh vien da muon sach\n";
            _borrowPay.search_borrow_pay_namestudent(borrow_pay);
            break;
        case 7:
            cout << "[7] : Exit\n";
            return;
        default:
            cout << "\t\tNhap sai lua chon!\n";
            break;
        }
        if (chon != 7)
        {
            cout << endl;
            cout << "=========================" << endl;
            cout << "=> Nhan Enter de tro ve !";
```

```
            cin.ignore();
            cin.get();
        }
    } while (chon != 7);
}
```

- Chức năng: Hiển thị menu tìm kiếm và thực hiện các chức năng tìm kiếm sách theo mã, tên sách, thể loại, sinh viên theo mã, tên sinh viên, phiếu mượn theo tên sinh viên.
- Độ phức tạp: O(n), với n là số lượng sách, sinh viên hoặc phiếu mượn, tùy thuộc vào chức năng cụ thể.

**6. menu6():**

```
void menu6()
    {
        int chon;
        do
        {
            system("cls");
            cout << "======================= MENU 6 ==========================\n";
            cout << "==                                                     ==\n";
            cout << "==        1.Thong ke so sinh vien trong thu vien.      ==\n";
            cout << "==        2.Thong ke so dau sach trong thu vien.       ==\n";
            cout << "==        3.Thong ke so phieu muon.                    ==\n";
            cout << "==        4.Exit.                                      ==\n";
            cout << "=========================================================\n";
            cout << "=> Moi chon chuc nang: ";
            cin >> chon;
            cout << "=========================" << endl;
            switch (chon)
            {
            case 1:
                cout << "[1] : Thong ke so sinh vien trong thu vien\n";
                cout << "Hien dang co ";
                cout << students.size() << endl;
                cout << " sinh vien trong thu vien\n";
                break;
            case 2:
                cout << "[2] : Thong ke so dau sach trong thu vien: ";
                cout << books.size() << endl;
                break;
            case 3:
                cout << "[3] : Thong ke so phieu muon : ";
```

```
            cout << borrow_pay.size() << endl;
            break;
        case 4:
            cout << "[4] : Exit\n";
            return;
        default:
            cout << "\t\tNhap sai lua chon!\n";
            break;
        }
        if (chon != 4)
        {
            cout << endl;
            cout << "=========================" << endl;
            cout << "=> Nhan Enter de tro ve !";
            cin.ignore();
            cin.get();
        }
    } while (chon != 4);
}
```

- Chức năng: Hiển thị menu thống kê và thực hiện các chức năng thống kê số lượng sinh viên, đầu sách, phiếu mượn.
- Độ phức tạp: O(n), với n là số lượng sinh viên, sách hoặc phiếu mượn, tùy thuộc vào chức năng cụ thể.

**7. menu_admin():**

```
void menu_admin() {
    // lúc bắt đầu chương trình sẽ lấy all data từ file push vào vector
    // để đồng bộ vector vs file
        _books.readinputData_Sach(books);
        _borrowPay.inputData_PhieuMuon(borrow_pay);
        int chon;
        do
        {
            system("cls");
            flagv2 = false;
            cout << "\t     Chuong Trinh Quan Ly Thu Vien." << endl;
            cout << "====================== MENU ADMIN ======================\n";
            cout << "==                                                    ==\n";
            cout << "==        1.Quan ly sach.                             ==\n";
            cout << "==        2.Quan ly phieu muon sach.                  ==\n";
            cout << "==        3.Quan ly cac ban doc trong thu vien.       ==\n";
            cout << "==        4.Sap xep.                                  ==\n";
```

```
        cout << "==        5.Tim kiem.                                    ==\n";
        cout << "==        6.Thong ke.                                    ==\n";
        cout << "==        7.Exit.                                        ==\n";
        cout << "==                                                       ==\n";
        cout << "===========================================================\n";
        cout << "=> Moi chon chuc nang: ";
        cin >> chon;
        cout << "=========================" << endl;
        switch (chon)
        {
        case 1:
            flagv2 = true;
            cout << "[1] : Quan ly sach\n";
            menu1();
            break;
        case 2:
            flagv2 = true;
            cout << "[2] : Quan ly phieu muon sach\n";
            menu2();
            break;
        case 3:
            flagv2 = true;
            cout << "[3] : Quan ly danh sach ban doc trong thu vien\n";
            menu3();
            break;
        case 4:
            flagv2 = true;
            cout << "[4] : Sap xep sach\n";
            menu4();
            break;
        case 5:
            flagv2 = true;
            cout << "[5] : Tim kiem sach\n";
            menu5();
            break;
        case 6:
            flagv2 = true;
            cout << "[6] : Thong ke\n";
            menu6();
            break;
        case 7:
            flagv2 = true;
            cout << "[7] : Exit\n";
            cout << "\n\t\tXin chao va hen gap lai !\n";
            break;
```

```
            default:
                cout << "\t\tNhap sai lua chon!\n";
                break;
            }
            if (chon != 7 && flagv2 == false)
            {
                cout << endl;
                cout << "=========================" << endl;
                cout << "=> Nhan Enter de tro ve !";
                cin.ignore();
                cin.get();
            }
        } while (chon != 7);
    }
```

- Chức năng: Hiển thị menu chính và chọn chức năng từ các menu con.
- Độ phức tạp: O(n), với n là số lượng sách, sinh viên hoặc phiếu mượn, tùy thuộc vào chức năng cụ thể.

**8. menu_user(string username, string password, string fullname):**

```
void menu_user(string username, string password, string fullname) {
    // lúc bắt đầu chương trình sẽ lấy all data từ file push vào vector
    // để đồng bộ vector vs file
        _books.readinputData_Sach(books);
        int chon;
        do {
            system("cls");
            cout << "\t     Chuong Trinh Quan Ly Thu Vien." << endl;
            cout << "  Hello : " << fullname << endl;
            int cnt_phieuquanhan = _borrowPay.check_phieuquahan_motsinhvien(borrow_pay,
            username);
            if (cnt_phieuquanhan)
                cout << "[Warning] : Ban co " << cnt_phieuquanhan << " quyen sach chua
            tra cho thu vien!\n";
            cout << "====================== MENU USER ======================\n";
            cout << "==                                                   ==\n";
            cout << "==        1.In toan bo sach co trong thu vien.       ==\n";
            cout << "==        2.Muon sach.                               ==\n";
            cout << "==        3.Tra sach.                                ==\n";
            cout << "==        4.In cac phieu muon sach cua ban.          ==\n";
            cout << "==        5.Tim kiem sach theo ten.                  ==\n";
            cout << "==        6.Tim kiem theo the loai sach ban muon doc.  ==\n";
            cout << "==        7.Sap xep sach theo tang dan gia tien.     ==\n";
```

```
        cout << "==        8.Dang xuat.                                    ==\n";
        cout << "==                                                        ==\n";
        cout << "============================================================\n";
        cout << "=> Moi chon chuc nang: ";
        cin >> chon;
        cout << "=========================" << endl;
        switch (chon)
        {
        case 1:
            cout << "[1] : In toan bo sach co trong thu vien\n";
            _books.ToanBoSach(books);
            break;
        case 2:
            cout << "[2] : Muon sach\n";
            _borrowPay.MuonSach(books, borrow_pay);
            break;
        case 3:
            cout << "[3] : Tra sach\n";
            _borrowPay.TraSach(books, borrow_pay);
            break;
        case 4:
            cout << "[4] : In cac phieu muon sach cua ban\n";
            _borrowPay.inphieumuon_motsinhvien(borrow_pay, username);
            break;
        case 5:
            cout << "[5] : Tim kiem sach theo ten\n";
            _books.search_book_to_name(books);
            break;
        case 6:
            cout << "[6] : Tim kiem sach theo the loai ban muon doc\n";
            _books.search_book_to_category(books);
            break;
        case 7:
            cout << "[7] : Sap xep sach theo tang dan gia tien\n";
            _books.arrange_book_amount(books);
            cout << "\t\tDa sap xep thanh cong!\n";
            break;
        case 8:
            cout << "[8] : Dang xuat\n";
            cout << "\n\t\tXin chao va hen gap lai !\n";
            break;
        default:
            cout << "\t\tNhap sai lua chon!\n";
            break;
        }
```

```
            if (chon != 8)
            {
                cout << endl;
                cout << "=========================" << endl;
                cout << "=> Nhan Enter de tro ve !";
                cin.ignore();
                cin.get();
            }
        } while (chon != 8);
    }
```

- Chức năng: Hiển thị menu chính dành cho sinh viên với các chức năng như in toàn bộ sách, mượn sách, trả sách, in phiếu mượn, tìm kiếm sách, sắp xếp sách. Nếu có phiếu quá hạn, hiển thị cảnh báo.
- Độ phức tạp: O(n), với n là số lượng sách hoặc số lượng phiếu mượn, tùy thuộc vào chức năng cụ thể.

### 3.11 main():

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
bool flag = false;   // check xem web đang ở tư cách admin hay người dùng
bool flagv2 = false; // check đã vào các mục menu con hay chưa, nếu vào rồi mà ra đây thì trước đó đã bấm Exit
bool ok2 = false;    // check xem đã đăng nhập vào web hay chưa
bool ok = false;     // check đăng ký
class StringManipulator···
//-----------------------------------------------
class Date···

class Sach : public StringManipulator···
//-------------------------------------------------------------------------------------------
class BanDoc : public StringManipulator···
//-------------------------------------------------------------------------------------------
class PhieuMuon : public BanDoc, public Sach, public Date···
int PhieuMuon::nNgayPlus(int nN)···

int PhieuMuon::nThangPlus(int nN)···

int PhieuMuon::nNamPlus(int nN)···
bool PhieuMuon::ktrNamNhuan()···
//-----------------------------------------------------------------
class List_Books : public StringManipulator···
//-----------------------------------------------------------------
class List_students : public StringManipulator···
class List_BorrowPay : public List_Books···
//-----------------------------------------------------------------------
class app···
class Login_Signup : public app, public StringManipulator···
int main()
{
    Login_Signup key;
    key.admin_login();
    return 0;
}
```

## IV. TÀI LIỆU THAM KHẢO:

[1] : Slide, video bài giảng về data structures & algorithm và oop trên lms link: *https://lms.utc.edu.vn*

[2] : C++, data structures and algorithm : *https://www.geeksforgeeks.org/*

[3] : Một số mã nguồn trên youtube, github của các a/c đi trước.

[4] : Các bài học online trên youtube:

Kênh: freeCodeCamp.org link: https://www.youtube.com/@freecodecamp

## V. DEMO CHẠY CODE.

- Đường link dẫn đến video demo chương trình bằng console:

# Phần B: Bài 23.

## I. ĐỀ BÀI :

Cây tìm kiếm nhị phân để tạo cấu trúc dữ liệu tập hợp (set)

1. Cài đặt cấu trúc dữ liệu set là một cấu trúc dữ liệu trừu tượng tổ chức trên cây tìm kiếm nhị phân các phần tử lưu trữ không có trùng lặp (nếu đã có không thêm vào cây)
2. Viết hàm main thực hiện thao tác sau
    a. Nhập dữ liệu các tập số nguyên A từ file A.txt và tập B từ file B.txt, tập C từ file C.txt
    b. Thêm lần lượt các phần tử của tập B vào tập A để được tập hợp hợp của A và B
    c. Bớt dần các phần tử của C khỏi A

## II. PHÂN TÍCH BÀI TOÁN

### a. Yêu cầu.

- Cài đặt set trong thư viện stl bằng cây nhị phân tìm kiếm.
- SET<int> A,B,C; (lấy dữ liệu đọc từ file)
- Thêm các phần tử của tập B vào A.
- Bớt dần các phần tử của C khỏi A.

### b. Tìm hiểu và phân tích.

### b.1 Tìm hiểu set trong thư viện stl

**Set** là một loại associative containers để lưu trữ các phần tử không bị trùng lặp

(unique elements), và các phần tử này chính là các khóa (keys).

- Khi duyệt set theo iterator từ begin đến end, các phần tử của set sẽ tăng dần về giá trị số or theo thứ tự từ điển(xâu).Set được thực hiện giống như cây tìm kiếm nhị phân (Binary search tree).

### b.2 Tìm hiểu Cây nhị phân tìm kiếm

**Cây nhị phân** là 1 cấu trúc dữ liệu mà mỗi node cha có tối đa 2 node con thường gọi là node con bên trái và node con bên phải.

Mỗi node trên cây nhị phân gồm 3 thành phần: Phần dữ liệu, địa chỉ node con bên trái, địa chỉ node con bên phải.

**Cây nhị phân tìm kiếm - Binary search tree (BST)** là cây nhị phân tìm kiếm sự xuất hiện của một phần tử trên cây với độ phức tạp logN. Điều kiện để thoả mãn cây BST là:

- Tất cả các node con thuộc cây con bên trái nhỏ hơn node gốc.

- Tất cả các node con thuộc cây con bên phải lớn hơn node gốc.

- Cây con bên trái và cây con bên phải cũng là cây BST.

## c. Xác định các lớp, các thuộc tính, các phương thức của lớp, chức năng.

### c.1 CLASS Node.

#### c.1.1 Chức năng

- Xây dựng một cây tìm kiếm nhị phân với các phương thức cơ bản như khởi tạo node mới, tìm kiếm, chèn, xoá một node.

#### c.1.2 Thuộc tính

- int val; // giá trị của node

- Node *left; // Lưu địa chỉ của node con bên trái

- Node *right; // Lưu địa chỉ của node con bên phải

#### c.1.3 Phương thức

- makeNode(T x):
  Chức năng : Tạo ra một node mới cho cây.

- findBST(Node<T> *root, T key):
  Chức năng : Tìm kiếm trong cây BST node có giá trị key.

- add(Node<T> *&root, T key):
  Chức năng : Chèn một node mới có giá trị key vào cây tìm kiếm nhị phân.
- minNode(Node<T> *root):
  Chức năng : Tìm node nhỏ nhất trong cây tìm kiếm nhị phân phục vụ cho deleteNode.
- deleteNode(Node<T> *root, T key):
  Chức năng : Xóa node có giá trị key khỏi cây tìm kiếm nhị phân.
- CLEAR(Node<T> *&root):
  Chức năng Xóa cây tìm kiếm nhị phân và giải phóng tất cả bộ nhớ đã cấp phát.

## c.2 CLASS VECTOR

### c.2.1 Chức năng:

Xây dựng Vector trong C++ có dynamic array (mảng động) có khả năng tự động thay đổi kích thước khi một phần tử được thêm vào với việc lưu trữ của chúng sẽ được vùng chứa tự động xử lý.

**Nhưng hơn hết chức năng chính là phục vụ cho việc duyệt SET (STL) và truy cập, trả về định dạng iterator.**

### c.2.2 Thuộc tính

- int n, cap; //n : size, cap : capacity
- T *buf; //buf : buffer

### c.2.3 Phương thức

- auto_resize(int new_cap):

  Chức năng : Mở rộng dung lượng của mảng động
- VECTOR():

  Chức năng : Hàm khởi tạo mặc định, tạo một đối tượng VECTOR mới.
- ~VECTOR():

  Chức năng : Hàm huỷ, giải phóng bộ nhớ khi đối tượng VECTOR bị huỷ
- swap(T &a, T &b):

  Chức năng : Hoán đổi giá trị của các phần tử a và b trong mảng VECTOR
- size():

  Chức năng : Trả về kích thước hiện tại của VECTOR

- push_back(T x):

  Chức năng : thêm vào cuối VECTOR

- clear():

  Chức năng : Xóa tất cả các phần tử trong VECTOR

- operator[](T index):

  Chức năng : Trả về giá trị buf[index] ở vị trí = index.

- typedef T* iterator: định nghĩa kiểu dữ liệu con trỏ đến T.

- begin():

  Chức năng : Trả về một iterator cho phần tử đầu tiên trong VECTOR

- end():

  Chức năng : Trả về một iterator cho phần tử có vị trí sau phần tử cuối cùng trong VECTOR

- add_value_toVECTOR(Node<T> *root):

  Chức năng : Duyệt cây tìm kiếm nhị phân root và thêm các value của mỗi node vào VECTOR v

- partition(int l, int r):

  Chức năng : Phân vùng VECTOR v bằng sơ đồ phân vùng Lomuto: so sánh các phần tử trong VECTOR với phần tử pivot, ta sẽ thu được 1 danh sách đã được sắp xếp phục vụ cho quick_sort.

- quick_sort(int l, int r):

  Chức năng : Sắp xếp VECTOR v bằng thuật toán quick sort: chia nhỏ danh sách gốc thành 2 danh sách con thông qua phương pháp so sánh partition ta sẽ thu được 1 danh sách gồm n phần tử nhỏ hơn or bằng pivot và 1 danh sách lớn hơn pivot.

- setting_lower_bound(int l, int r, int key):

  Chức năng : Thuật toán tìm kiếm nhị phân trả về vị trí của phần tử đầu tiên trong VECTOR v lớn hơn hoặc bằng key, hoặc -1 nếu không tồn tại phần tử đó.

- setting_upper_bound(int l, int r, T key):

  Chức năng : Thuật toán tìm kiếm nhị phân trả về vị trí của phần tử đầu tiên trong VECTOR v lớn hơn key, hoặc -1 nếu không tồn tại phần tử đó

### c.3 CLASS SET

#### c.3.1: Chức năng.

Set được cài đặt bằng cây tìm kiếm nhị phân (Binary search tree), mặc định của set là sử dụng phép toán less, khi duyệt set theo iterator từ begin đến end, các phần tử của set sẽ tăng dần theo phép toán so sánh.

#### c.3.2: Thuộc tính.

- Node<T> *root = nullptr;
- int n;

#### c.3.3: Phương thức.

- SET(): Hàm khởi tạo mặc định tạo một đối tượng SET mới.
- size():

  Chức năng : Trả về kích thước hiện tại của SET
- empty():

  Chức năng : Trả về true nếu tập hợp trống và ngược lại.
- insert(T x):

  Chức năng : Chèn phần tử x vào tập hợp
- clear():

  Chức năng : Xóa tất cả các phần tử trong SET
- erase(T x):

  Chức năng : Xóa phần tử x khỏi SET
- count(T x):

  Chức năng : Trả về số lần xuất hiện của phần tử x trong SET
- begin():

  Chức năng : Trả về iterator trỏ đến phần tử đầu tiên của SET, định nghĩa phương thức này để có thể sử dụng vòng lặp
- end():

  Chức năng : Trả về iterator trỏ đến phần tử sau phần tử cuối cùng trong SET, định nghĩa phương thức này để sử dụng vòng lặp
- find(T x):

Chức năng : Trả về itarator trỏ đến phần tử cần tìm kiếm. Nếu không tìm thấy thì itarator trỏ về "end" của SET.

– erase(iterator it):

Chức năng : Xóa phần tử theo iterator

– lower_bound(T key):

Chức năng : trả về iterator đến vị trí phần tử bé nhất mà không bé hơn (lớn hơn hoặc bằng) key (dĩ nhiên là theo phép so sánh), nếukhông tìm thấy trả về vị trí "end" của SET.

– upper_bound(T key):

Chức năng : Trả về iterator đến vị trí phần tử bé nhất mà lớn hơn key, nếu không tìm thấy trả về vị trí "end" của SET.

## III. CÀI ĐẶT CÁC LỚP VÀ HÀM MAIN BẰNG C++

### a. CLASS Node.

```
template <class T>
class Node
{
public:
    int val;
    Node *left;  // Lưu địa chỉ của node con bên trái
    Node *right; // Lưu địa chỉ của node con bên phải
```

### a.1 makeNode(T x):

```
Node<T> *makeNode(T x)
{
    Node<T> *newNode = new Node<T>(); // Allocate memory for the new node
    newNode->val = x;
    newNode->right = newNode->left = NULL;
    return newNode;
}
```

### a.2 findBST(Node<T> *root, T key):

```
// 1. Thao tác tìm kiếm
bool findBST(Node<T> *root, T key)
{
    if (root == NULL)
        return false;
```

```
        if (root->val == key)
            return true;
        else if (root->val < key)
            return findBST(root->right, key);
        else
            return findBST(root->left, key);
    }
```

### a.3 add(Node<T> *&root, T key):

```
    // 2.Thao tác chèn
    void add(Node<T> *&root, T key)
    {
        if (root == NULL)
        {
            root = makeNode(key);
            return;
        }
        if (root->val < key)
            add(root->right, key);
        else
            add(root->left, key);
    }
```

### a.4 minNode(Node<T> *root):

```
    Node<T> *minNode(Node<T> *root){
        // Find node con min_element and > root
        Node<T> *tmp = root;
        while (tmp->left != NULL && tmp != NULL)
        {
            tmp = tmp->left;
        }
        return tmp;
    }
```

### a.5 deleteNode(Node<T> *root, T key):

```
     Node<T> *deleteNode(Node<T> *root, T key)
     {
         if (root == NULL)
             return root;
         if (root->val > key)
         {
             root->left = deleteNode(root->left, key);
```

```
        }
        else if (root->val < key)
            root->right = deleteNode(root->right, key);
        else
        {
            if (root->left == NULL)
            {
                Node<T> *temp = root->right;
                delete root; // thu hoi vung root nay
                return temp;
            }
            else if (root->right == NULL)
            {
                Node<T> *temp = root->left;
                delete root;
                return temp;
            }
            else
            {
                Node<T> *temp = minNode(root);
                root->val = temp->val;
                root->right = deleteNode(root->right, temp->val);
            }
        }
        return root;
    }
```

### a.6 CLEAR(Node<T> *&root):

```
void CLEAR(Node<T> *&root) // giải phóng bộ nhớ và dọn dẹp các tài nguyên cây
{
    if (root != nullptr)
    {
        CLEAR(root->left);
        CLEAR(root->right);
        delete root;
        root = nullptr; // tránh sử dụng địa chỉ bộ nhớ đã xóa
    }
}
```

### b. CLASS VECTOR:

```
template <class T>
class VECTOR
{
```

```
        int n, cap; // n : size, cap : capacity
        T *buf;     // buffer
```

### b.1 auto_resize(int new_cap):

```
// mở rộng kích thước của một mảng động.
    void auto_resize(int new_cap)
    {
        if (new_cap < cap)
            return;
        T *tem = buf;
        cap = new_cap;
        buf = new T[cap]; // tạo mảng mới
        for (int i = 0; i < n; i++)
            buf[i] = tem[i];
        if (tem)
            delete[] tem;
    }
```

### b.2 VECTOR():

```
    VECTOR()
        {
            n = cap = 0;
            buf = NULL;
        }
```

### b.3 ~VECTOR():

```
    ~VECTOR()
        {
            if (buf)
                delete[] buf;
        }
```

### b.4 swap(T &a, T &b):

```
    void swap(T &a, T &b)
        {
            T tmp = a;
            a = b;
            b = tmp;
        }
```

### b.5 size():

```
int size()
    {
        return n;
    }
```

### b.6 push_back(T x):

```
void push_back(T x)
 {
    if (n == cap)
        auto_resize(cap * 2 + 1);
        buf[n++] = x;
    }
```

### b.7 clear():

```
 void clear()
     {
         n = 0;
     }
```

### b.8 operator[](T index):

```
T &operator[](T index)
    {
        return buf[index];
    }
```

### b.9 begin() & end():

```
typedef T *iterator;
    iterator begin() // trả về địa chỉ
    {
        return buf;
    }
    iterator end()
    {
        return buf + n;
    }
```

### b.10 add_value_toVECTOR(Node<T> *root):

```
// Duyệt cây rồi thêm value các node vào VECTOR
     void add_value_toVECTOR(Node<T> *root)
     {
```

```
            if (root != NULL)
            {
                add_value_toVECTOR(root->left);
                push_back(root->val);
                add_value_toVECTOR(root->right);
            }
        }
```

### b.11 partition(int l, int r):

```
        T partition(int l, int r)
            {
                int pivot = buf[r];
                int i = l - 1;
                for (int j = l; j < r; ++j)
                {
                    if (buf[j] <= pivot)
                    {
                        ++i;
                        swap(buf[j], buf[i]);
                    }
                }
                ++i;
                swap(buf[i], buf[r]);
                return i;
            }
```

### b.12 quick_sort(int l, int r):

```
        void quick_sort(int l, int r)
            {
                if (l < r)
                {
                    int p = partition(l, r);
                    quick_sort(l, p - 1);
                    quick_sort(p + 1, r);
                }
            }
```

### b.13 setting_lower_bound(int l, int r, int key):

```
       T setting_lower_bound(int l, int r, int key)
            {
                T res = -1;
                while (l <= r)
```

```
        {
            int mid = (l + r) / 2;
            if (buf[mid] >= key)
            {
                res = mid;
                r = mid - 1;
            }
            else
                l = mid + 1;
        }
        return res;
    }
```

### b.14 setting_upper_bound(int l, int r, T key):

```
    T setting_upper_bound(int l, int r, T key)
        {
            T res = -1;
            while (l <= r)
            {
                int mid = (l + r) / 2;
                if (buf[mid] > key)
                {
                    res = mid;
                    r = mid - 1;
                }
                else
                    l = mid + 1;
            }
            return res;
        }
```

## c. CLASS SET

```
template <class T>
class SET
{
public:
    Node<T> *root = nullptr;
    int n;
```

### c.1 size():

```
int size()
    {
        return n;
    }
```

### c.2 empty():

```
bool empty()
    {
        return n == 0;
    }
```

### c.3 insert(T x):

```
void insert(T x)
    {
        if (!root || !root->findBST(root, x))
        {
            ++n;
            root->add(root, x);
        }
    }
```

### c.4 clear():

```
void clear()
    {
        root->CLEAR(root);
        n = 0;
    }
```

### c.5 erase(T x):

```
void erase(T x)
    {
        if (root->findBST(root, x))
        {
            --n;
            root = root->deleteNode(root, x);
        }
    }
```

### c.6 count(T x):

```
int count(T x)
    {
        if (root->findBST(root, x))
            return 1;
```

```
            return 0;
        }
```

### c.7 begin():

```
    iterator begin()
        {
            v.clear();
            v.add_value_toVECTOR(root);
            return v.begin();
        }
```

### c.8 end():

```
    iterator end()
        {
            return v.end();
        }
```

### c.9 find(T x):

```
    //Tìm kiếm key và trả về kiểu iterator
        iterator find(T key)
        {
            v.clear();
            v.add_value_toVECTOR(root);
            for (int i = 0; i < v.size(); ++i)
            {
                if (v[i] == key)
                    return v.begin() + i;
            }
            return v.end();
        }
```

### c.10 erase(iterator it):

```
// cùng tên hàm nhưng tham số truyền vào nó khác nên compiler sẽ biết mà phân biệt
    void erase(iterator it)
    {
        erase(*it);
        --n;
    }
```

### c.11 lower_bound(T key):

```
iterator lower_bound(T key)
    {
        v.clear();
```

```
        v.add_value_toVECTOR(root);
        v.quick_sort(0, n - 1);
        int x = v.setting_lower_bound(0, v.size() - 1, key);
        if (x != -1)
            return v.begin() + x;
        else
            return v.end();
    }
```

### c.12 upper_bound(T key):

```
    iterator upper_bound(T key)
        {
            v.clear();
            v.add_value_toVECTOR(root, v);
            v.quick_sort(0, n - 1);
            int x = v.setting_upper_bound(0, v.size() - 1, key);
            if (x != -1)
                return v.begin() + x;
            else
                return v.end();
        }
```

**c.4 main().**

Ngoài ra còn có hàm đọc file text.

```
void readFile(string file, SET<int> &A)
{
    int a;
    ifstream File(file);
    if (!File.is_open())
    {
        cout << "Error: " << file << " not found" << endl;
        return;
    }
    while (!File.eof())
    {
        File >> a;
        A.insert(a);
        File.ignore(1, ' ');
    }
    File.close();
}
```

```cpp
// SETTING SET(STL) = BINARY SEARCH TREE
#include <bits/stdc++.h>
using namespace std;
template <class T>
class Node ···

template <class T>
class VECTOR ···
template <class T>
class SET ···
void readFile(string file, SET<int> &A) ···
int main()
{
    SET<int> A, B, C;
    readFile("A.txt", A);//1 2 3 4 5 8
    readFile("B.txt", B);//5 4 11 56 0
    readFile("C.txt", C);//20 2 10 3 7 12 4
    for (int x : B)
    {
        A.insert(x);
    }
    for (int x : A)
    {
        if (C.count(x))
        {
            A.erase(x);
        }
    }
    for (int x : A)
        cout << x << " ";//0 1 5 8 11 56
}
```

## IV. PHÂN TÍCH THỜI GIAN CHẠY CỦA TỪNG PHƯƠNG THỨC CÓ TRONG CÁC LỚP.

### a. CLASS Node.

**a.1 makeNode(T x):**

- Có 3 phép gán giá trị cho node
- Có 1 phép cấp phát bộ nhớ để tạo một node mới
- Có 1 phép trả về node đã được tạo

=> Tổng số phép toán cơ sở: 5 phép toán

=> Độ phức tạp: O(5)

**a.2 findBST(Node<T> *root, T key):**

- Có 2 lệnh if tương đương với có 2 phép toán

- Có 2 phép trả về kiểu boolean : 2 phép toán

- Goị đệ quy cho cây con trái và cây con phải: 2 phép toán, O(log(n/2))

=> Tổng số phép toán cơ sở: 6 phép toán

Trường hợp xấu nhất xảy ra khi khóa không có trong BST và hàm đi qua toàn bộ cây từ gốc đến lá. Trong trường hợp này, số lượng cuộc gọi đệ quy bằng với chiều cao của BST.

=> Với cây nhị phân tìm kiếm có n lá thì chiều cao của cây khoảng log(n). Độ phức tạp: O(log (n))

**a.3 add(Node<T> *&root, T key):**

- Có 2 lệnh if tương đương với có 2 phép toán

- Có 1 phép gán giá trị cho node

- Goị đệ quy cho cây con trái và cây con phải: 2 phép toán, O(log(n/2))

=> Tổng số phép toán cơ sở: 5 phép toán

Tương tự như trên:

=> Với cây nhị phân tìm kiếm có n lá thì chiều cao của cây khoảng log(n). Độ phức tạp: O(log (n))

**a.4 minNode(Node<T> *root):**

- Có 2 lệnh if tương đương với có: 2 phép toán

- Có 2 phép gán giá trị: 1 phép toán

- Có 1 phép trả về node đã được sao chép: 1 phép toán

=> Tổng số phép toán cơ sở: 4 phép toán

Trong trường hợp xấu nhất, khi BST bị lệch trái, hàm phải duyệt toàn bộ cây con bên trái, dẫn đến độ phức tạp thời gian là O(n) trong đó n là chiều cao của cây, có thể lớn bằng O(n) đối với BST không cân bằng. Tuy nhiên, trong BST cân bằng, chiều cao xấp xỉ log n, dẫn đến độ phức tạp về thời gian tổng thể là O (log n).

**a.5 deleteNode(Node<T> *root, T key):**

Tương tự như trên:

=> Với cây nhị phân tìm kiếm có n lá thì chiều cao của cây khoảng log(n).
Độ phức tạp: O(log (n))

**a.6 CLEAR(Node<T> *&root):**

- Có 1 phép gán giá trị cho node : 1 phép toán
- Có 1 lệnh if tương đương với : 1 phép toán
- Có 1 phép xoá node: 1 phép toán.
- Goị đệ quy cho cây con trái và cây con phải: 2 phép toán, O(log(n/2))

=> Tổng số phép toán cơ sở: 5 phép toán

=> Độ phức tạp: O(log (n)).

**b. CLASS VECTOR:**

**b.1 auto_resize(int new_cap):**

- Một câu lệnh điều kiện: 1 phép toán
- Cấp phát động: n phép toán
- Vòng lặp có n phép toán
- Lệnh gán trong for: n phép toán
- Lệnh gán giá trị: 1 phép toán
- Giải phóng bộ nhớ: n phép toán

=> Tổng: 4n+2 phép toán cơ sở

=> Độ phức tạp: O(n)

**b.2 VECTOR():**

- Có 3 câu lệnh gán giá trị : 3 phép toán

=> Độ phức tạp: O(1).

**b.3 ~VECTOR():**

- Xoá, giải phóng bộ nhớ : n phép toán

=> Độ phức tạp: O(n).

**b.4 swap(T &a, T &b):**

- Có 3 câu lệnh gán giá trị : 3 phép toán

=> Độ phức tạp: O(1).

**b.5 size():**

- Trả về giá trị size: 1 phép toán

=> Độ phức tạp: O(1)

**b.6 push_back(T x):**

- Câu lệnh tang giá trị lên 1 đơn vị: 1 phép toán

- Gọi hàm expand() : O(n)

- Gán dữ liệu mới vào vector: 1 phép toán

=> Độ phức tạp: O(n)

**b.7 clear():**

- 1 phép gán n = 0; => O(1).

**b.8 operator[](T index):**

- Có 1 phép trả về giá trị buf [index]. => O(1)

**b.9 begin() & end():**

- Có 1 phép trả về địa chỉ(iterator) => O(1).

**b.10 add_value_toVECTOR(Node<T> *root):**

- Có 1 phép push_back. => O(n).

- Goị đệ quy cho cây con trái và cây con phải: => O(log(n)) || O(n).

=> Vậy tính theo trường hợp xấu nhất nên độ phức tạp của hàm là O(n).

**b.11 partition(int l, int r):**

=> Độ phức tạp: O(n).

**b.12 quick_sort(int l, int r):**

=> Độ phức tạp: O(n * log(n)).

**b.13 setting_lower_bound(int l, int r, int key):**

- Tìm kiếm nhị phân => O(log(n)).

**b.14 setting_upper_bound(int l, int r, T key):**

- Tìm kiếm nhị phân => O(log(n)).

**c. CLASS SET**

**c.1 size():**

- Có 1 phép trả về n: 1 phép toán => O(1).

**c.2 empty():**

- Có 1 phép trả về kiểu boolean: 1 phép toán => O(1).

**c.3** **insert(T x):**

- Có 1 phép tăng giá trị lên 1 đơn vị: 1 phép toán
- Gọi đệ quy findBST kiểm tra cây có tồn tại node x: 1 phép toán, O(log(n))
- Gọi đệ quy add để thêm vào cây node x: 1 phép toán, O(log(n))

=> Độ phức tạp: O(log(n)).

**c.4** **clear():**

- Có 1 phép gán giá trị : 1 phép toán
- Gọi đệ quy CLEAR để xoá cây: 1 phép toán, O(log(n))

=> Độ phức tạp: O(log(n)).

**c.5** **erase(T x):**

- Có 1 phép tăng giá trị lên 1 đơn vị: 1 phép toán
- Gọi đệ quy findBST kiểm tra cây có tồn tại node x: 1 phép toán, O(log(n))
- Gọi đệ quy deleteNode để thêm vào cây node x: 1 phép toán, O(log(n))

=> Độ phức tạp: O(log(n)).

**c.6** **count(T x):**

- Có 2 phép trả về : 2 phép toán.
- Gọi đệ quy findBST kiểm tra cây có tồn tại node x: 1 phép toán, O(log(n))

=> Độ phức tạp: O(log(n)).

**c.7** **begin():**

- hàm clear của VECTOR có độ phức tạp : O(1)
- Hàm add_value_toVECTOR : O(n)
- Có 1 phép trả về: O(1)

=> Độ phức tạp: O(n).

**c.8** **end():**

- Có 1 phép trả về: => Độ phức tạp: O(1).

**c.9** **find(T x):**

- Vòng lặp n lần: n phép toán
- Có 2 câu lệnh trả về: 2 phép toán

=> Độ phức tạp: O(n).

**c.10 erase(iterator it):**

- Có 1 câu lệnh trả về : O(1)

- Gọi hàm erase với tham số đầu vào là giá trị: O(log(n)).

=> Độ phức tạp: O(log(n)).

**c.11 lower_bound(T key):**

- hàm v.clear() của VECTOR: O(1)

- hàm v.add_value_toVECTOR() : O(n)

- hàm v.quick_sort() của VECTOR: O(n*log(n))

- hàm v.setting_lower_bound() của VECTOR: O(log(n)).

- Có 1 phép gán giá trị: 1 phép toán.

- Có 2 phép trả về kiểu iterator: 2 phép toán.

=> Độ phức tạp: O(n*log(n)).

**c.12 upper_bound(T key):**

- hàm v.clear() của VECTOR: O(1)

- hàm v.add_value_toVECTOR() : O(n)

- hàm v.quick_sort() của VECTOR: O(n*log(n))

- hàm v.setting_upper_bound() của VECTOR: O(log(n)).

- Có 1 phép gán giá trị: 1 phép toán.

- Có 2 phép trả về kiểu iterator: 2 phép toán.

=> Độ phức tạp: O(n*log(n)).

## V. TÀI LIỆU THAM KHẢO

[1] : Slide, video bài giảng về môn data structures and algorithm và oop trên lms link: *https://lms.utc.edu.vn*

[2] : c++, data structures and algorithm : *https://www.geeksforgeeks.org/*

[3] : Source code của thầy Tích.