NĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

BÀI TẬP THỰC HÀNH 3

GVHD: Nguyễn Ngọc Quí

Sinh viên thực hiện: Trần Nam Phong - 23521172

Tp. Hồ Chí Minh, 04/2024

IT002 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

……, ngày…….....tháng……năm 2024

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

1. Xây dựng lớp phân số 6

1.1. Class Diagram của lớp PhanSo 6

1.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp PhanSo trong file PhanSo.h 6

1.3. Các phương thức của class PhanSo được xây dựng trong file PhanSo.cpp 7

1.4. Gọi các phương thức trong hàm main(), đặt trong file main.cpp 11

1.5. Kết quả khi chạy chương trình 13

2. Xây dựng lớp số phức 13

2.1. Class Diagram của lớp SoPhuc 13

2.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp SoPhuc trong file SoPhuc.h 14

2.3. Các phương thức của class SoPhuc được xây dựng trong file SoPhuc.cpp 14

2.4. Gọi các phương thức trong hàm main(), đặt trong file main.cpp 17

2.5. Kết quả khi chạy chương trình 18

3. Xây dựng lớp thời gian 18

3.1. Class diagram của lớp ThoiGian 18

3.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp ThoiGian trong file ThoiGian.h 19

3.3. Các phương thức của class ThoiGian được xây dựng trong file ThoiGian.cpp 19

3.4. Gọi các phương thức trong hàm main(), đặt trong file main.cpp 24

3.5. Kết quả khi chạy chương trình 26

4. Xây dựng lớp ngày tháng năm 26

4.1. Class Diagram của lớp NgayThangNam 27

4.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp NgayThangNam trong file NgayThangNam.h 27

4.3. Các phương thức của class NgayThangNam được xây dựng trong file NgayThangNam.cpp 28

4.4. Gọi các phương thức trong hàm main(), đặt trong file main.cpp 40

4.5. Kết quả khi chạy chương trình 42

DANH MỤC BẢNG

Bảng 1. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp PhanSo 6
Bảng 2. Các phương thức của class PhanSo được xây dựng trong file PhanSo.cpp 8
Bảng 3. Gọi các phương thức của class PhanSo trong hàm main() 11
Bảng 4. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp SoPhuc 14
Bảng 5. Các phương thức của class SoPhuc được xây dựng trong file SoPhuc.cpp 15
Bảng 6. Gọi các phương thức của class SoPhuc trong hàm main() 17
Bảng 7. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp ThoiGian..... 19
Bảng 8. Các phương thức của class ThoiGian được xây dựng trong file ThoiGian.cpp 20
Bảng 9. Gọi các phương thức của class ThoiGian trong hàm main()..... 24
Bảng 10. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp NgayThangNam 27
Bảng 11. Các phương thức của class NgayThangNam được xây dựng trong file NgayThangNam.cpp 29
Bảng 12. Gọi các phương thức của class NgayThangNam trong hàm main() 40

DANH MỤC HÌNH ẢNH

Hình 1. Class Diagram của lớp PhanSo 6
Hình 2. Kết quả khi chạy chương trình Bài 1 13
Hình 3. Class Diagram của lớp SoPhuc 13
Hình 4. Kết quả của chương trình Bài 2 18
Hình 5. Class diagram của lớp ThoiGian 18
Hình 6. Kết quả của chương trình Bài 3 26
Hình 7. Class Diagram của lớp NgayThangNam 27
Hình 8. Kết quả của chương trình Bài 4 42

NỘI DUNG BÀI LÀM

1. Xây dựng lớp phân số

• Thuộc tính: iTu, iMau

• Phương thức: PhanSo(), PhanSo(int Tu, int Mau)

• Thực hiện các phương thức operator: +, -, *, /, ==, !=, >=, <=, >, <, >>, <<

Yêu cầu: Thực hiện xây dựng lớp, vẽ class diagram và khai báo các thuộc tính, phương thức. Viết nội dung vào các phương thức đã khai báo. Gọi các phương thức trong hàm main().

1.1. Class Diagram của lớp PhanSo

Hình 1. Class Diagram của lớp PhanSo

1.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp PhanSo trong file *PhanSo.h*

Bảng 1. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp PhanSo

```
#pragma once
#include<iostream>
```

```
#include<cmath>
#include<algorithm>
using namespace std;

class PhanSo
{
private:
	int iTu, iMau;
public:
	PhanSo() {};
	PhanSo(int tu, int mau);
	PhanSo rutgon(PhanSo a);
	PhanSo operator+ (PhanSo a);
	PhanSo operator- (PhanSo a);
	PhanSo operator* (PhanSo a);
	PhanSo operator/ (PhanSo a);
	bool operator== (PhanSo a);
	bool operator!= (PhanSo a);
	bool operator>= (PhanSo a);
	bool operator<= (PhanSo a);
	bool operator> (PhanSo a);
	bool operator< (PhanSo a);
	friend istream& operator>>(istream& is, PhanSo& a);
	friend ostream& operator<<(ostream& os, PhanSo a);
};
```

1.3. Các phương thức của class PhanSo được xây dựng trong file *PhanSo.cpp*

- Input: nhập phân số từ bàn phím
- Hướng giải quyết
  - Rút gọn: Chia cả tử và mẫu cho ước chung lớn nhất(của tử và mẫu)
  - operator+ (PhanSo a): a.iTu = iTu * a.iMau + a.iTu * iMau; a.iMau *= iMau và sau đó rút gọn
  - Các phương thức operator- (PhanSo a), operator* (PhanSo a), operator/ (PhanSo a) giải quyết tương tự theo các quy tắc toán học cơ bản
  - Phương thức operator==(PhanSo a):
    - a.iTu *= iMau; iTu *= a.iMau
    - Nếu (a.iTu == iTu) trả về true ngược lại trả về false

- Phương thức operator!=(PhanSo a) tương tự phương thức operator==(PhanSo a) nhưng ngược lại
- Phuong thuc operator>=(PhanSo a) :
  - a.iTu *= iMau; iTu *= a.iMau;
  - Nếu (a.iTu <= iTu) trả về true, ngược lại trả về false
- Các phương thức operator<= (PhanSo a), operator> (PhanSo a), operator< (PhanSo a) giải quyết tương tự theo các quy tắc toán học cơ bản
- Phương thức operator>>(istream& is, PhanSo& a): is >> a.iTu >> a.iMau và trả về is
- Phương thức operator<<(ostream& os, PhanSo a)
  - Nếu (a.iMau == 0) thì cout << a.iTu ngược lại os << a.iTu << "/" << a.iMau;
  - Cuối cùng trả về os

Bảng 2. Các phương thức của class PhanSo được xây dựng trong file PhanSo.cpp

```
#include<iostream>
#include<cmath>
#include<algorithm>
#include "PhanSo.h"

using namespace std;

int ucln(int a, int b)
{
	if (a == 0 || b == 0) {
		return a + b;
	}
	while (a != b) {
		if (a > b) {
			a -= b; // a = a - b
		}
		else {
			b -= a;
		}
	}
	return a;
}
```

```
PhanSo PhanSo::rutgon(PhanSo a)
{
	//int x= __gcd(a.iMau, a.iTu);
	int x = ucln(a.iTu,a.iMau);
	//int x = 1;
	a.iMau /= x;
	a.iMau /= x;
	return a;
}
PhanSo::PhanSo(int tu, int mau)
{
	iTu = tu;
	iMau = mau;
}
PhanSo PhanSo::operator+ (PhanSo a) {
	a.iTu = iTu * a.iMau + a.iTu * iMau;
	a.iMau *= iMau;
	return rutgon(a);
}
PhanSo PhanSo::operator- (PhanSo a) {
	a.iTu = a.iTu * iMau - iTu * a.iMau;
	a.iMau *= iMau;
	return rutgon(a);
}
PhanSo PhanSo::operator* (PhanSo a) {
	a.iTu *= iTu;
	a.iMau *= iMau;
	return rutgon(a);
}
PhanSo PhanSo::operator/ (PhanSo a) {
	a.iTu *= iMau;
	a.iMau *= iTu;
	return rutgon(a);
}
bool PhanSo::operator==(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
```

```
	if (a.iTu == iTu)
		return true;
	return false;
}
bool PhanSo::operator!=(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
	if (a.iTu != iTu)
		return true;
	return false;
}
bool PhanSo::operator>=(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
	if (a.iTu <= iTu)
		return true;
	return false;
}
bool PhanSo::operator<=(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
	if (a.iTu >= iTu)
		return true;
	return false;
}
bool PhanSo::operator>(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
	if (a.iTu < iTu)
		return true;
	return false;
}
bool PhanSo::operator<(PhanSo a) {
	a.iTu *= iMau;
	iTu *= a.iMau;
	if (a.iTu > iTu)
		return true;
```

```
	return false;
}
istream& operator>>(istream& is, PhanSo& a) {
	is >> a.iTu >> a.iMau;
	return is;
}
ostream& operator<<(ostream& os, PhanSo a) {
	if (a.iMau == 0)
		cout << a.iTu;
	else
		os << a.iTu << "/" << a.iMau;

	return os;
}
```

1.4. Gọi các phương thức trong hàm main(), đặt trong file *main.cpp*

Bảng 3. Gọi các phương thức của class PhanSo trong hàm main()

```
#include "PhanSo.h"
#include<iostream>
#include<cmath>
#include<algorithm>
using namespace std;

int main()
{
	PhanSo ans;
	PhanSo a(2,3);
	PhanSo b(1,1);
	cout << "Phan so a:" << a<<'\n';
	cout << "Phan so b:" << b<<'\n';
	cout << "Tong hai phan so:";
	ans = b + a;
	cout << ans << endl;
	cout << "Hieu hai phan so:";
	ans = a - b;
	cout << ans << endl;
	cout << "Tich hai phan so:";
```

```
	ans = a * b;
	cout << a << endl;
	cout << "Thuong hai phan so:";
	ans = a / b;
	cout << ans << endl;
	cout << "Phan so a bang phan so b:";
	if (a == b)
		cout << "YES";
	else
		cout << "NO";
	cout << "\nPhan so a khong bang phan so b:";
	if (a != b)
		cout << "YES";
	else
		cout << "NO";
	cout << "\nPhan so a lon hon hoac bang phan so b:";
	if (a >= b)
		cout << "YES";
	else
		cout << "NO";
	cout << "\nPhan so a be hon hoac bang phan so b:";
	if (a <= b)
		cout << "YES";
	else
		cout << "NO";
	cout << "\nPhan so a lon hon phan so b:";
	if (a > b)
		cout << "YES";
	else
		cout << "NO";
	cout << "\nPhan so a be hon phan so b:";
	if (a < b)
		cout << "YES";
	else
		cout << "NO";
}
```

1.5. Kết quả khi chạy chương trình

```
Phan so a:2/3
Phan so b:1/1
Tong hai phan so:5/3
Hieu hai phan so:1/3
Tich hai phan so:2/3
Thuong hai phan so:3/2
Phan so a bang phan so b:NO
Phan so a khong bang phan so b:YES
Phan so a lon hon hoac bang phan so b:NO
Phan so a be hon hoac bang phan so b:YES
Phan so a lon hon phan so b:NO
Phan so a be hon phan so b:YES
```

Hình 2. Kết quả khi chạy chương trình Bài 1

2. Xây dựng lớp số phức

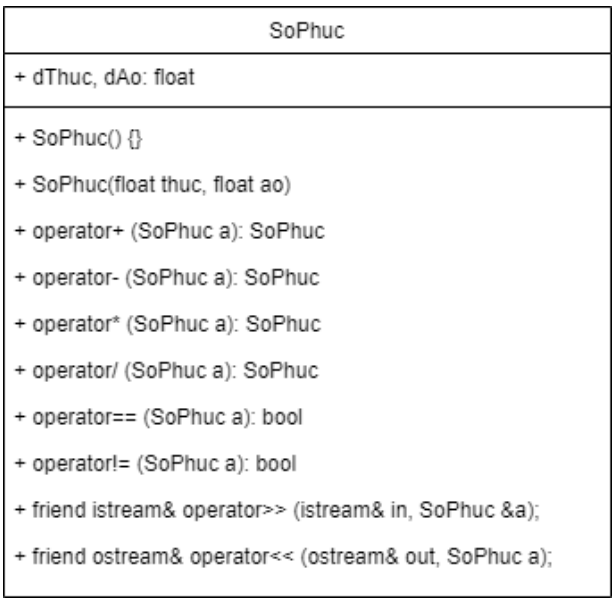
• Thuộc tính: dThuc, dAo

• Phương thức: SoPhuc(), SoPhuc (int thuc, int ao)

• Thực hiện các phương thức operator: +, -, *, /, ==, !=, >>, <<

Yêu cầu: Thực hiện xây dựng lớp, vẽ class diagram và khai báo các thuộc tính, phương thức. Viết nội dung vào các phương thức đã khai báo. Gọi các phương thức trong hàm main().

2.1. Class Diagram của lớp SoPhuc



Hình 3. Class Diagram của lớp SoPhuc

2.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp SoPhuc trong file *SoPhuc.h*

Bảng 4. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp SoPhuc

```
#pragma once
#include<bits/stdc++.h>
using namespace std;

class SoPhuc {
private:
	float dThuc, dAo;
public:
	SoPhuc() {};
	SoPhuc(float thuc, float ao);
	SoPhuc operator+ (SoPhuc a);
	SoPhuc operator- (SoPhuc a);
	SoPhuc operator* (SoPhuc a);
	SoPhuc operator/ (SoPhuc a);
	bool operator== (SoPhuc a);
	bool operator!= (SoPhuc a);
	friend istream& operator>> (istream& in, SoPhuc &a);
	friend ostream& operator<< (ostream& out, SoPhuc a);
};
```

2.3. Các phương thức của class SoPhuc được xây dựng trong file *SoPhuc.cpp*

- Input: Nhập lần lượt phần thực và phần ảo của hai số phức
- Output: In ra màn hình kết quả cộng, trừ, nhân, chia và so sánh hai số phức
- Hướng giải quyết:
  - Phương thức SoPhuc::operator+ (SoPhuc a):
    - kq.dThuc = dThuc + a.dThuc;
    - kq.dAo = dAo + a.dAo
    - Tương tự cho phương thức SoPhuc::operator- (SoPhuc a).
  - Phương thức SoPhuc::operator* (SoPhuc a):
    - kq.dThuc = dThuc * a.dThuc - dAo * a.dAo
    - kq.dAo = dAo * a.dThuc + dThuc * a.dAo
  - Phương thức SoPhuc::operator/ (SoPhuc a):
    - float mau = a.dThuc * a.dThuc + a.dAo * a.dAo;
    - kq.dThuc =(float) (dThuc * a.dThuc + dAo * a.dAo) / mau;
    - kq.dAo =(float) (dAo * a.dThuc - dThuc * a.dAo) / mau;
  - Phương thưc SoPhuc::operator== (SoPhuc a): Ta so sánh phần thực và phần ảo tương ứng của hai số phức với nhau, nếu bằng nhau thì trả về true,

ngược lại trả về false. Đối với phương thưc SoPhuc::operator!= (SoPhuc a), ta sẽ trả về ngược lại.

Bảng 5. Các phương thức của class SoPhuc được xây dựng trong file SoPhuc.cpp

```
#include"SoPhuc.h"
#include<bits/stdc++.h>
using namespace std;

SoPhuc::SoPhuc(float thuc, float ao) {
	dThuc = thuc;
	dAo = ao;
}
istream& operator>> (istream& in, SoPhuc& a) {
	cout << "Nhap phan thuc: ";
	in >> a.dThuc;
	cout << "Nhap phan ao: ";
	in >> a.dAo;
	return in;
}

ostream& operator<< (ostream& out, SoPhuc a) {
	if (a.dAo < 0) {
		out << a.dThuc << a.dAo << "i";
	}
	else if (a.dAo == 0) {
		out << a.dThuc;
	}
	else {
		out << a.dThuc << "+" << a.dAo << "i";
	}
	return out;
}

SoPhuc SoPhuc::operator+ (SoPhuc a) {
	SoPhuc kq;
	kq.dThuc = dThuc + a.dThuc;
	kq.dAo = dAo + a.dAo;
```

```
	return kq;
}

SoPhuc SoPhuc::operator- (SoPhuc a) {
	SoPhuc kq;
	kq.dThuc = dThuc - a.dThuc;
	kq.dAo = dAo - a.dAo;
	return kq;
}

SoPhuc SoPhuc::operator* (SoPhuc a) {
	SoPhuc kq;
	kq.dThuc = dThuc * a.dThuc - dAo * a.dAo;
	kq.dAo = dAo * a.dThuc + dThuc * a.dAo;
	return kq;
}
// (a+bi)(c-di) = (ac+ bd) + (-ad + bc)i
SoPhuc SoPhuc::operator/ (SoPhuc a) {
	SoPhuc kq;
	float mau = a.dThuc * a.dThuc + a.dAo * a.dAo;
	kq.dThuc =(float) (dThuc * a.dThuc + dAo * a.dAo) /
mau;
	kq.dAo =(float) (dAo * a.dThuc - dThuc * a.dAo) /
mau;
	return kq;
}

bool SoPhuc::operator== (SoPhuc a) {
	if (dThuc == a.dThuc && dAo == a.dAo) {
		return true;
	}
	else return false;
}

bool SoPhuc::operator!= (SoPhuc a) {
	if (dThuc == a.dThuc && dAo == a.dAo) {
		return false;
```

```
        }
        else return true;
}
```

2.4. Gọi các phương thức trong hàm main(), đặt trong file *main.cpp*

Bảng 6. Gọi các phương thức của class SoPhuc trong hàm main()

```
#include"SoPhuc.h"
#include<bits/stdc++.h>
using namespace std;

int main() {
        SoPhuc a, b;
        cout << "Nhap phan so thu nhat:  \n";
        cin >> a;
        cout << "Nhap phan so thu hai:  \n";
        cin >> b;
        cout << "Tong hai so phuc la: " << a + b << endl;
        cout << "Hieu hai so phuc la: " << a - b << endl;
        cout << "Tich hai so phuc la: " << a * b << endl;
        cout << "Thuong hai so phuc la: " << a / b << endl;
        cout << "Hai so phuc bang nhau: ";
        if (a == b) {
                cout << "YES";
        }
        else {
                cout << "NO";
        }
        cout << endl;
        cout << "Hai so phuc khac nhau: ";
        if (a != b) {
                cout << "YES";
        }
        else {
                cout << "NO";
        }
        return 0;
```

```
}
```

2.5. Kết quả khi chạy chương trình

```
Nhap phan so thu nhat:
Nhap phan thuc: 3
Nhap phan ao: 1
Nhap phan so thu hai:
Nhap phan thuc: 2
Nhap phan ao: -1
Tong hai so phuc la: 5
Hieu hai so phuc la: 1+2i
Tich hai so phuc la: 7-1i
Thuong hai so phuc la: 1+1i
Hai so phuc bang nhau: NO
Hai so phuc khac nhau: YES
```

Hình 4. Kết quả của chương trình Bài 2

3. Xây dựng lớp thời gian

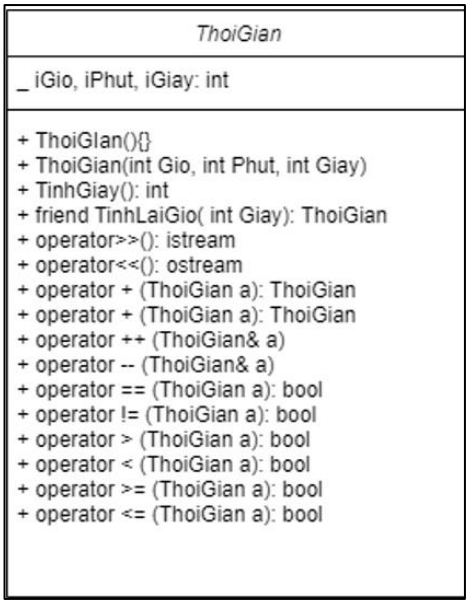
• Thuộc tính: iGio, iPhut, iGiay

• Phương thức: ThoiGian(), ThoiGian (int Gio, int Phut, int Giay), TinhGiay(), TinhLaiGio(int Giay)

• Thực hiện các phương thức operator: +(int Giay), -(int Giay), +(ThoiGian a), -(ThoiGian a), ++, --, ==, !=, >=, <=, >, <, >>, <<

Yêu cầu: Thực hiện xây dựng lớp, vẽ class diagram và khai báo các thuộc tính, phương thức. Viết nội dung vào các phương thức đã khai báo. Gọi các phương thức trong hàm main()

3.1. Class diagram của lớp ThoiGian



Hình 5. Class diagram của lớp ThoiGian

3.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp ThoiGian trong file *ThoiGian.h*

Bảng 7. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp ThoiGian

```
#include<iostream>
using namespace std;
class ThoiGian
{
private:
    int iGio, iPhut, iGiay;

public:
    ThoiGian();
    ThoiGian(int Gio, int Phut, int Giay);

    int TinhGiay();
    friend ThoiGian TinhLaiGio(int Giay);
    friend istream &operator>>(istream &is, ThoiGian &a);
    friend ostream &operator<<(ostream &os, ThoiGian a);
    friend ThoiGian operator+(ThoiGian a, int Giay);
    friend ThoiGian operator-(ThoiGian a, int Giay);
    ThoiGian operator + (ThoiGian a);
    ThoiGian operator - (ThoiGian a);
    friend void operator++(ThoiGian& a);
    friend void operator--(ThoiGian& a);
    bool operator == (ThoiGian a);
    bool operator != (ThoiGian a);
    bool operator > (ThoiGian a);
    bool operator < (ThoiGian a);
    bool operator >= (ThoiGian a);
    bool operator <= (ThoiGian a);
};
```

3.3. Các phương thức của class ThoiGian được xây dựng trong file ThoiGian.cpp

- Input:
    - Nhập lần lượt giờ, phút, giây của 1 lớp ThoiGian
    - Nhập số giây.

- Output:
    - In ra giờ, phút, giây của lớp ThoiGian vừa nhập và một lớp ThoiGian tự đặt.
    - Số giây của ThoiGian thứ nhất.
    - Thời gian của số giây vừa nhập.
    - Giá trị của thời gian sau khi cộng, trừ số giây.
    - Giá trị của hai thời gian cộng, trừ với nhau.
    - Xuất giá trị của thời gian thứ nhất sau khi tắng, giảm một giây.
    - Kết quả của các phép Bool (==, !=, <, >, <=, >=) của hai thời gian.
- Hướng giải quyết:
    - Phương thức tính giây: return (iGio * 60 + iPhut) * 60 + iGiay;
    - Phương thức tính lại giờ:
        - Phut = Giay / 60; Giay -= Phut * 60; Gio = Phut / 60; Phut -= Gio * 60;
        - ThoiGian x(Gio, Phut, Giay);
        - return x;
    - Phương thức cộng thêm giây (tương tự với trừ): Tính giây của ThoiGian, sau đó cộng thêm giây và tính lại giờ.
    - Phương thức cộng Thời gian (Tương tự với trừ Thời gian): Tính giây của hai thời gian, sau đó cộng cả hai giây lại và tính lại giờ.
    - Phương thức operator++: cộng một giây (Tương tự với --)
    - Phương thức operator > (Tương tự các operator khác): Tính giây và trả về kết quả so sánh 2 giây.

Bảng 8. Các phương thức của class ThoiGian được xây dựng trong file ThoiGian.cpp

```
#include <iostream>
using namespace std;

ThoiGian::ThoiGian()
{
    iGio = 0;
    iPhut = 0;
    iGiay = 0;
}
ThoiGian::ThoiGian(int Gio, int Phut, int Giay)
{
    iGio = Gio;
    iPhut = Phut;
```

```
    iGiay = Giay;
}



bool ThoiGian::operator==(ThoiGian a)
{
    return a.iGiay==this->iGiay && a.iGio==this->iGio &&
a.iPhut==this->iPhut;
}
bool ThoiGian::operator!=(ThoiGian a)
{
    return a.iGiay!=this->iGiay || a.iGio!=this->iGio ||
a.iPhut!=this->iPhut;
}
bool ThoiGian::operator > (ThoiGian a)
{
    if(this->iGio != a.iGio){
        return this->iGio > a.iGio;
    }
    if(this->iPhut != a.iPhut){
        return this->iPhut > a.iPhut;
    }
    if(this->iGiay != a.iGiay){
        return this->iGiay > a.iGiay;
    }
    return false;
}
bool ThoiGian::operator < (ThoiGian a)
{

    if(*this > a || *this==a) return false;
    return true;
}
istream &operator>>(istream &is, ThoiGian &a)
{
    cout << "Gio: ";
    is >> a.iGio;
```

```
    cout << "Phut: ";
    is >> a.iPhut;
    cout << "Giay: ";
    is >> a.iGiay;
    while (a.iGiay < 0 || a.iGiay >= 60 || a.iPhut < 0 ||
a.iPhut >= 60 || a.iGio < 0 || a.iGio >= 24)
    {
        cout << "Vui long nhap lai" << endl;
        cout << "Gio: ";
        is >> a.iGio;
        cout << "Phut: ";
        is >> a.iPhut;
        cout << "Giay: ";
        is >> a.iGiay;
    }
    return is;
}
ostream &operator<<(ostream &os, ThoiGian a)
{
    if (a.iGio < 10)
        os << 0;
    os << a.iGio << ":";
    if (a.iPhut < 10)
        os << 0;
    os << a.iPhut << ":";
    if (a.iGiay < 10)
        os << 0;
    os << a.iGiay << endl;
    return os;
}


int ThoiGian::TinhGiay()
{
    return (iGio * 60 + iPhut) * 60 + iGiay;
}
ThoiGian TinhLaiGio(int Giay)
```

```
{
    int Phut = Giay / 60;
    Giay -= Phut * 60;
    int Gio = Phut / 60;
    Phut -= Gio * 60;
    ThoiGian x(Gio, Phut, Giay);
    return x;
}
ThoiGian operator+(ThoiGian a, int Giay)
{
    a.iGiay+=Giay;
    int x=a.TinhGiay();
    a=TinhLaiGio(x);
    return a;
}
ThoiGian operator-(ThoiGian a, int Giay)
{
    Giay*=-1;
    return a+Giay;
}
ThoiGian ThoiGian::operator+(ThoiGian a)
{
    a.iGiay+=this->iGiay;
    a.iPhut+=this->iPhut;
    a.iGio+=this->iGio;
    int x=a.TinhGiay();
    return TinhLaiGio(x);
}
ThoiGian ThoiGian::operator-(ThoiGian a)
{
    a.iGiay=this->iGiay-a.iGiay;
    a.iPhut=this->iPhut-a.iPhut;
    a.iGio=this->iGio-a.iGio;
    int x=a.TinhGiay();
    return TinhLaiGio(x);
}
void operator++(ThoiGian& a)
```

```
{
    a=a+1;
}
void operator--(ThoiGian& a)
{
    a=a-1;
}
bool ThoiGian::operator >= (ThoiGian a)
{
    if(*this==a || *this > a) return true;
    return false;
}
bool ThoiGian::operator <= (ThoiGian a)
{
    if(*this==a || *this<a) return true;
    return false;
}
```

3.4. Gọi các phương thức trong hàm main(), đặt trong file *main.cpp*

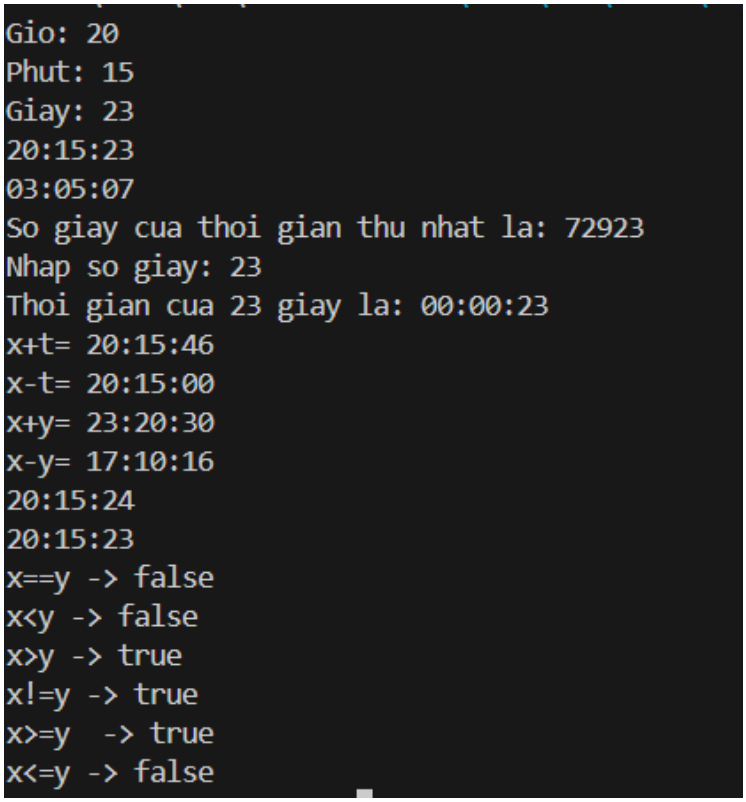
Bảng 9. Gọi các phương thức của class ThoiGian trong hàm main()

```
#include <iostream>
using namespace std;
#include "ThoiGian.h"
#include "ThoiGian.cpp"

int main()
{
    ThoiGian x, y(3, 5, 7);
    cin >> x;
    cout << x << y;
    cout << "So giay cua thoi gian thu nhat la: " <<
x.TinhGiay() << endl;
    int t;
    cout << "Nhap so giay: ";
    cin >> t;
    cout << "Thoi gian cua " << t << " giay la: ";
    cout << TinhLaiGio(t);
```

```
    cout <<"x+t= "<< x + t;
    cout <<"x-t= "<< x - t;
    cout <<"x+y= "<< x + y;
    cout <<"x-y= "<< x - y;
    ++x;
    cout << x;
    --x;
    cout << x;
    cout << "x==y -> ";
    if (x == y)
        cout << "true" << endl;
    else
        cout << "false" << endl;
    cout<<"x<y -> ";
    if(x<y) cout<<"true"<<endl;
    else cout<<"false"<<endl;
    cout<<"x>y -> ";
    if(x>y) cout<<"true"<<endl;
    else cout<<"false"<<endl;
    cout << "x!=y -> ";
    if (x != y)
        cout << "true" << endl;
    else
        cout << "false" << endl;
    cout << "x>=y  -> ";
    if (x >= y)
        cout << "true" << endl;
    else
        cout << "false" << endl;
    cout << "x<=y -> ";
    if (x <= y)
        cout << "true" << endl;
    else
        cout << "false" << endl;
}
```

3.5. Kết quả khi chạy chương trình



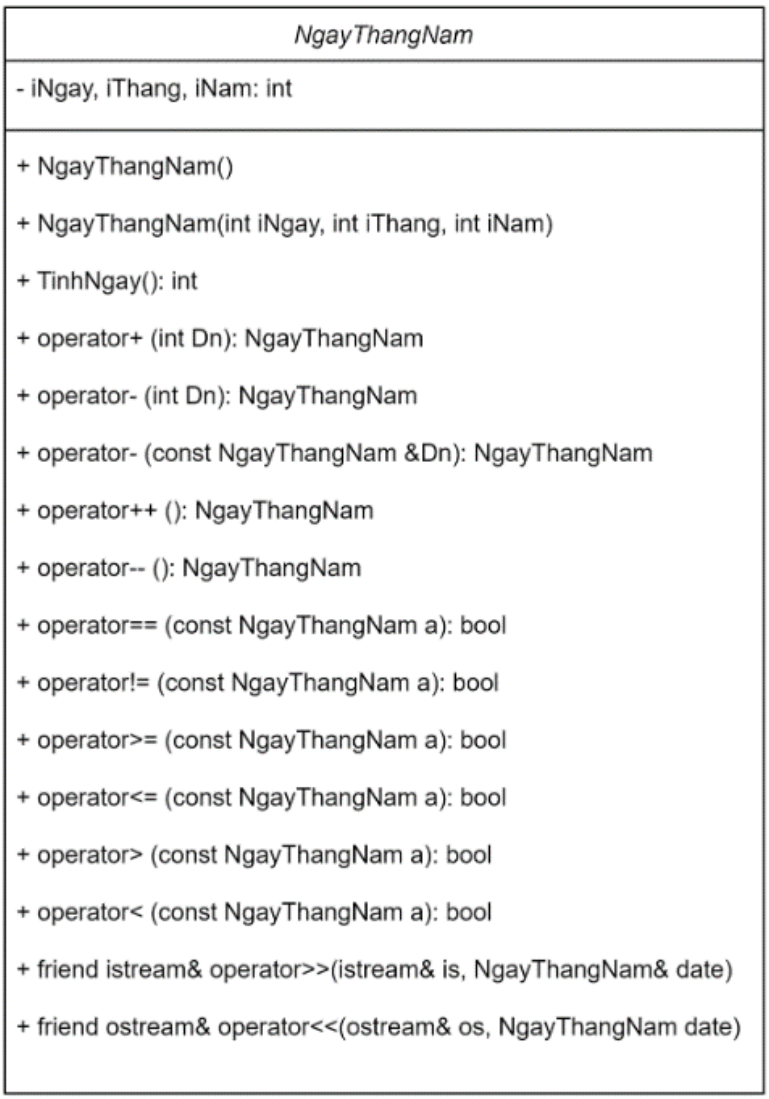
Hình 6. Kết quả của chương trình Bài 3

4. Xây dựng lớp ngày tháng năm

- Thuộc tính: iNgay, iThang, iNam
- Phương thức: NgayThangNam(), NgayThangNam (int Nam, int Thang, int Ngay = 1), TinhNgay()
- Thực hiện các phương thức operator: +(int ngay), -(int ngay), -(NgayThangNam a), ++, --, ==, !=, >=, <=, >, <, >>, <<

Yêu cầu: Thực hiện xây dựng lớp, vẽ class diagram và khai báo các thuộc tính, phương thức. Viết nội dung vào các phương thức đã khai báo. Gọi các phương thức trong hàm main()

4.1. Class Diagram của lớp NgayThangNam



Hình 7. Class Diagram của lớp NgayThangNam

4.2. Thực hiện xây dựng lớp, khai báo các thuộc tính, phương thức của lớp NgayThangNam trong file NgayThangNam.h

Bảng 10. Xây dựng lớp, khai báo các thuộc tính, phương thức của lớp NgayThangNam

```
#pragma once

#include <iostream>

class NgayThangNam
{
private:
    int iNgay;
    int iThang;
```

```
    int iNam;


public:
    NgayThangNam();
    NgayThangNam(int iNgay, int iThang, int iNam);


    int TinhNgay();


    friend std::istream &operator>>(std::istream &is,
NgayThangNam &date);
    friend std::ostream &operator<<(std::ostream &os, const
NgayThangNam &date);


    NgayThangNam operator+(int Dn);
    NgayThangNam operator-(int Dn);
    NgayThangNam operator-(const NgayThangNam &Dn);


    bool operator>(const NgayThangNam &a);
    bool operator<(const NgayThangNam &a);
    bool operator<=(const NgayThangNam &a);
    bool operator>=(const NgayThangNam &a);
    bool operator==(const NgayThangNam &a);
    bool operator!=(const NgayThangNam &a);


    NgayThangNam &operator++();
    NgayThangNam &operator--();
};
```

4.3. Các phương thức của class NgayThangNam được xây dựng trong file NgayThangNam.cpp

- Input: Nhập vào ngày ban đầu, ngày cần sử dụng để thực hiện tính toán, ngày cần sử dụng để cộng/trừ, và ngày cần so sánh
- Output: trả về các kết quả sau khi thực hiện tính toán và phép so sánh.
- Hướng giải quyết:
    - Phương thức NgayThangNam(): Constructor mặc định
    - Phương thức NgayThangNam(int iNgay, int iThang, int iNam): Constructor khởi tạo đối tượng.

- Phương thức TinhNgay(): dùng mảng lưu số ngày của các tháng, kiểm tra năm nhuận để xác định số ngày của tháng 2. Cộng dồn số ngày từ đầu năm đến tháng hiện tại -> tổng số ngày.
- Phương thức operator>>(istream &is, NgayThangNam &date): gán lần lượt cho iNgay, iThang, iNam của đối tượng.
- Phương thức operator<<(ostream &os, const NgayThangNam &date): Xuất giá trị iNgay, iThang, iNam theo định dạng ngay/thang/nam.
- Phương thức operator+(int Dn): Tạo một đối tượng NgayThangNam mới với giá trị ban đầu là ngày hiện tại. Cộng Dn vào iNgay của đối tượng mới. Nếu iNgay vượt quá số ngày của tháng, tăng iThang và điều chỉnh iNgay tương ứng. Nếu iThang vượt quá 12 cũng điều chỉnh lại.
- Phương thức operator-(int Dn): thực hiện phép trừ Dn ngày từ ngày hiện tại. Nếu iNgay nhỏ hơn 1, giảm iThang và điều chỉnh iNgay tương ứng. Nếu iThang nhỏ hơn 1, giảm iNam.
- Phương thức operator-(const NgayThangNam &Dn): Tìm NgayThangNam mới bằng cách trừ iNgay, iThang, iNam của ngày hiện tại cho đối tượng Dn. sau đó điều chỉnh lại.
- Phương thức operator>(const NgayThangNam &a): So sánh iNam của ngày hiện tại với iNam của a. Nếu bằng nhau, so sánh iThang. Nếu iThang cũng bằng nhau, so sánh iNgay. Trả về true nếu ngày hiện tại lớn hơn a.
- Phương thức operator<(const NgayThangNam &a): Tương tự như `operator>` nhưng trả về `true` nếu ngày hiện tại nhỏ hơn `a`.
- Phương thức operator<=(const NgayThangNam &a): Tương tự như `operator<' nhưng xét thêm điều kiện nếu cả 2 ngày bằng nhau -> trả về true
- Phương thức operator>=(const NgayThangNam &a): Tương tự như `operator>' nhưng xét thêm điều kiện nếu cả 2 ngày bằng nhau -> trả về true
- Phương thức operator==(const NgayThangNam &a): So sánh giữa 2 ngày. Trả về true nếu chúng bằng nhau.
- Phương thức operator!=(const NgayThangNam &a): Ngược lại với `operator==`.
- Phương thức operator++(): Tăng iNgay lên 1. Nếu iNgay vượt quá số ngày của tháng, tăng iThang lên 1 và điều chỉnh iNgay. Nếu iThang vượt quá 12, tăng iNam lên 1 và đặt lại iNgay, iThang bằng 1.
- Phương thức operator--(): Giảm iNgay đi 1. Nếu iNgay nhỏ hơn 1, giảm iThang đi 1 và đặt iNgay bằng số ngày của tháng trước đó. Nếu iThang nhỏ hơn 1, giảm iNam đi 1 và đặt iThang bằng 12.

Bảng 11. Các phương thức của class NgayThangNam được xây dựng trong file NgayThangNam.cpp

```
#include "NgayThangNam.h"


NgayThangNam::NgayThangNam() {}


NgayThangNam::NgayThangNam(int iNgay, int iThang, int iNam) :
iNgay(iNgay), iThang(iThang), iNam(iNam) {}
```

```
int NgayThangNam::TinhNgay()
{
    int soNgayTruoc[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if (iNam % 400 == 0 || (iNam % 4 == 0 && iNam % 100 != 0))
    {
        soNgayTruoc[2] = 29;
    }

    int tongSoNgay = iNgay;
    for (int i = 1; i < iThang; ++i)
    {
        tongSoNgay += soNgayTruoc[i];
    }

    return tongSoNgay;
}

std::istream &operator>>(std::istream &is, NgayThangNam &date)
{
    is >> date.iNgay;
    is >> date.iThang;
    is >> date.iNam;
    return is;
}

std::ostream &operator<<(std::ostream &os, const NgayThangNam &date)
{
    os << date.iNgay << "/" << date.iThang << "/" << date.iNam;
    return os;
}

NgayThangNam NgayThangNam::operator+(int Dn)
```

```
{
    NgayThangNam N;
    N.iNgay = this->iNgay + Dn;
    N.iThang = this->iThang;
    N.iNam = this->iNam;

    if (N.iNgay > 28 && N.iThang == 2)
    {
        if (N.iNam % 400 == 0 || (N.iNam % 4 == 0 && N.iNam %
100 != 0))
        {
            if (N.iNgay > 29)
            {
                ++N.iThang;
                N.iNgay -= 29;
            }
        }
        else
        {
            if (N.iNgay > 28)
            {
                ++N.iThang;
                N.iNgay -= 28;
            }
        }
    }

    if (N.iNgay > 30)
    {
        switch (this->iThang)
        {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
```

```
        case 12:
            ++N.iThang;
            N.iNgay -= 31;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            ++N.iThang;
            N.iNgay -= 30;
            break;
        }
    }

    if (N.iNgay == 31 && N.iThang == 12)
    {
        ++N.iNam;
        N.iNgay = 1;
        N.iThang = 1;
    }

    return N;
}

NgayThangNam NgayThangNam::operator-(int Dn)
{
    NgayThangNam N;
    N.iNgay = this->iNgay - Dn;
    N.iThang = this->iThang;
    N.iNam = this->iNam;

    if (N.iNgay < 1)
    {
        switch (this->iThang)
        {
        case 3:
```

```
            if (this->iNam % 400 == 0 || (this->iNam % 4 == 0
&& this->iNam % 100 != 0))
            {
                N.iThang = 2;
                N.iNgay += 29;
            }
            else
            {
                N.iThang = 2;
                N.iNgay += 28;
            }
            break;
        case 5:
        case 7:
        case 10:
        case 12:
            --N.iThang;
            N.iNgay += 30;
            break;
        case 1:
        case 2:
        case 4:
        case 6:
        case 8:
        case 9:
        case 11:
            --N.iThang;
            N.iNgay += 31;
            break;
        }
    }

    if (N.iThang < 1)
    {
        --N.iNam;
        N.iThang = 12;
    }
```

```
    return N;
}

NgayThangNam NgayThangNam::operator-(const NgayThangNam &Dn)
{
    NgayThangNam N;
    N.iNgay = this->iNgay - Dn.iNgay;
    N.iThang = this->iThang - Dn.iThang;
    N.iNam = this->iNam - Dn.iNam;

    if (N.iNgay < 0)
    {
        --N.iThang;
        switch (this->iThang)
        {
        case 2:
            if ((this->iNam % 400 == 0) || (this->iNam % 4 ==
0 && this->iNam % 100 != 0))
            {

                N.iNgay += 29;
            }
            else
            {

                N.iNgay += 28;
            }
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            N.iNgay += 30;
            break;
        default:
            N.iNgay += 31;
```

```
            break;
        }
    }

    if (N.iThang < 1)
    {
        N.iThang += 12;
        --N.iNam;
    }

    if (N.iNam < 1)
    {

        N.iNam = 1;
        N.iThang -= 12;
    }

    return N;
}

bool NgayThangNam::operator>(const NgayThangNam &a)
{
    if (a.iNam < iNam)
        return true;
    else if (a.iNam == iNam)
    {
        if (a.iThang < iThang)
            return true;
        else if (a.iThang == iThang)
        {
            if (a.iNgay < iNgay)
                return true;
            else
                return false;
        }
        else
            return false;
```

```
    }
    else
        return false;
}

bool NgayThangNam::operator<(const NgayThangNam &a)
{
    if (a.iNam > iNam)
        return true;
    else if (a.iNam == iNam)
    {
        if (a.iThang > iThang)
            return true;
        else if (a.iThang == iThang)
        {
            if (a.iNgay > iNgay)
                return true;
            else
                return false;
        }
        else
            return false;
    }
    else
        return false;
}
bool NgayThangNam::operator<=(const NgayThangNam &a)
{
    if (iNam < a.iNam)
        return true;
    else if (iNam == a.iNam && iThang < a.iThang)
        return true;
    else if (iNam == a.iNam && iThang == a.iThang && iNgay <=
a.iNgay)
        return true;

    return false;
```

```
}



bool NgayThangNam::operator>=(const NgayThangNam &a)
{
    if (iNam > a.iNam)
        return true;
    else if (iNam == a.iNam && iThang > a.iThang)
        return true;
    else if (iNam == a.iNam && iThang == a.iThang && iNgay >=
a.iNgay)
        return true;
    else
        return false;
}

bool NgayThangNam::operator==(const NgayThangNam &a)
{
    return (iNgay == a.iNgay && iThang == a.iThang && iNam ==
a.iNam);
}

bool NgayThangNam::operator!=(const NgayThangNam &a)
{
    return !(iNgay == a.iNgay && iThang == a.iThang && iNam
== a.iNam);
}

NgayThangNam &NgayThangNam::operator++()
{
    ++iNgay;

    switch (iThang)
    {
    case 2:
        if ((iNam % 4 == 0 && iNam % 100 != 0) || iNam % 400
== 0)
```

```
        {
            if (iNgay > 29)
            {
                ++iThang;
                iNgay = 1;
            }
        }
        else
        {
            if (iNgay > 28)
            {
                ++iThang;
                iNgay = 1;
            }
        }
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        if (iNgay > 30)
        {
            ++iThang;
            iNgay = 1;
        }
        break;
    default:
        if (iNgay > 31)
        {
            ++iThang;
            iNgay = 1;
            if (iThang > 12)
            {
                iThang = 1;
                ++iNam;
            }
        }
```

```
        break;
    }

    return *this;
}

NgayThangNam &NgayThangNam::operator--()
{
    --iNgay;

    if (iNgay <= 0)
    {
        --iThang;
        if (iThang == 0)
        {
            --iNam;
            iThang = 12;
        }

        int soNgayTruocDo = 31;
        switch (iThang)
        {
        case 2:
            if ((iNam % 4 == 0 && iNam % 100 != 0) || iNam %
400 == 0)
            {
                soNgayTruocDo = 29;
            }
            else
            {
                soNgayTruocDo = 28;
            }
            break;
        case 4:
        case 6:
        case 9:
        case 11:
```

```
            soNgayTruocDo = 30;
            break;
        }

        iNgay = soNgayTruocDo;
    }

    return *this;
}
```

4.4. Gọi các phương thức trong hàm main(), đặt trong file *main.cpp*

Bảng 12. Gọi các phương thức của class NgayThangNam trong hàm main()

```
#include <iostream>
#include "NgayThangNam.h"

int main()
{
    NgayThangNam a, b, c;
    std::cout << "Nhap ngay ban dau: " << std::endl;
    std::cin >> a;
    std::cout << "Nhap ngay can su dung de thuc hien tinh
toan: " << std::endl;
    std::cin >> b;
    std::cout << "Nhap ngay can cong/tru: " << std::endl;
    int day;
    std::cin >> day;

    NgayThangNam cong, tru, trudate;
    cong = a + day;
    tru = a - day;
    trudate = a - b;
    NgayThangNam d = a;
    NgayThangNam e = a;
    std::cout << "Ngay ban dau sau khi cong/tru " << day << "
ngay: " << std::endl;
    std::cout << "Sau khi cong: " << cong << std::endl;
    std::cout << "Sau khi tru: " << tru << std::endl;
```

```
    std::cout << "Ngay ban dau sau khi cong len 1 ngay: " <<
++d << std::endl;
    std::cout << "Ngay ban dau sau khi tru di 1 ngay: " << --
e << std::endl;
    std::cout << "Ngay ban dau sau khi tru cho " << b << " :"
<< trudate << std::endl;


    std::cout << "Cac phep so sanh:" << std::endl;
    std::cout << "Nhap ngay can so sanh: " << std::endl;
    NgayThangNam sosanh;
    std::cin >> sosanh;
    if (a > sosanh)
    {
        std::cout << a << " > " << sosanh << std::endl;
    }
    if (a < sosanh)
    {
        std::cout << a << " < " << sosanh << std::endl;
    }
    if (a >= sosanh)
    {
        std::cout << a << ">=" << sosanh << std::endl;
    }
    if (a <= sosanh)
    {
        std::cout << a << "<=" << sosanh << std::endl;
    }
    if (a == sosanh)
    {
        std::cout << a << " == " << sosanh << std::endl;
    }
    if (a != sosanh)
    {
        std::cout << a << " != " << sosanh << std::endl;
    }
    return 0;
}
```

4.5. Kết quả khi chạy chương trình

Hình 8. Kết quả của chương trình Bài 4

Hết

FILE CODE