

# BÁO CÁO BÀI TẬP 1

Môn học: Lập trình hướng đối tượng

Tên chủ đề: báo cáo bài tập 1

GVHD: Nguyễn Hữu Quyền

Ngày báo cáo: 23/03/2023

**Nhóm: 02**

## 1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: IT002.N28.2

STT	Họ và tên	MSSV	Email
1	Nguyễn Hải Phong	22521088	22521088@gm.uit.edu.vn
2	Hồ Trung Kiên	22520704	22520704@gm.uit.edu.vn

## 2. NỘI DUNG THỰC HIỆN:

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Câu hỏi 01	100%	Nguyễn Hải Phong
2	Câu hỏi 02	100%	Nguyễn Hải Phong
3	Câu hỏi 03	100%	Nguyễn Hải Phong
4	Câu hỏi 04	100%	Nguyễn Hải Phong
5	Câu hỏi 05	100%	Nguyễn Hải Phong
6	Câu hỏi 06	100%	Hồ Trung Kiên
7	Câu hỏi 07	100%	Hồ Trung Kiên
8	Câu hỏi 08	100%	Hồ Trung Kiên

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

# BÁO CÁO CHI TIẾT BÀI THỰC HÀNH 1

## 1. Câu hỏi 01

- Mô tả/mục tiêu: Viết chương trình nhập vào một phân số, rút gọn phân số và xuất kết quả.
- Các bước thực hiện/Phương pháp thực hiện:
  - Bước 1: Tạo một struct có tên PhanSo bao gồm 2 thành phần là tử số (tu) và mẫu số (mau). Tử số và mẫu số đều có kiểu dữ liệu là integer

```
struct PhanSo {  
    int tu, mau;  
};
```

- Bước 2: Trong hàm main tạo một biến a có kiểu dữ liệu là struct PhanSo. Sau đó thực hiện bước nhập tử và mẫu của phân số. Đồng thời viết theo câu lệnh kiểm tra xem phân số nhập vào có hợp lệ không bằng cách kiểm tra mẫu số đã khác 0 chưa, nếu chưa thì yêu cầu nhập lại. Nếu đã hợp lệ thì sẽ sử dụng hàm đã được viết trước là hàm RutGonPhanSo để tiến hành rút gọn phân số và xuất ra kết quả.

```
int main()  
{  
    PhanSo a;  
    cout << "Nhap phan so: ";  
    cin >> a.tu >> a.mau;  
    while (a.mau == 0) {  
        cout << "Vui long nhap lai: ";  
        cin >> a.tu >> a.mau;  
    }  
    RutGonPhanSo(a);  
    return 0;  
}
```

- Bước 3: Tạo 1 hàm kiểu dữ liệu void dùng để rút gọn phân số được nhập vào và xuất ra kết quả, đặt tên là RutGonPhanSo, trong đó tham số truyền vào là một biến a có kiểu dữ liệu là PhanSo. Trong hàm này, tiến hành đổi dấu cho tử và mẫu nếu như mẫu số là số âm và tử là số dương. Còn nếu tử và mẫu cùng âm thì cho tử và mẫu trở thành số dương. Sau đó sử dụng hàm gcd trong thư viện numeric để tìm ước chung lớn nhất cho tử số và mẫu số và gán cho biến sochia. Sau đó tiến hành chia tử và mẫu số cho biến sochia và tiến hành xuất ra phân số đã được rút gọn. Nếu kết quả có tử chia hết cho mẫu thì chỉ xuất ra tử số của phân số đó

```
void RutGonPhanSo(PhanSo &a) {  
    if (a.mau < 0 && a.tu < 0) {  
        a.tu = abs(a.tu);  
        a.mau = abs(a.mau);  
    }  
    else if (a.mau < 0 && a.tu >= 0) {  
        a.mau = abs(a.mau);  
        a.tu = (-1) * a.tu;  
    }  
    int sochia = gcd(abs(a.tu), abs(a.mau));  
    a.tu = a.tu / sochia;  
    a.mau = a.mau / sochia;  
    if (a.tu % a.mau == 0) cout << "Phan so da rut gon:" << a.tu;  
    else cout << "Phan so da rut gon: " << a.tu << "/" << a.mau;  
}
```

## 2. Câu hỏi 02

- Mô tả/mục tiêu: Viết chương trình nhập vào hai phân số, tìm phân số lớn nhất và xuất kết quả.
- Các bước thực hiện/ Phương pháp thực hiện:
  - Bước 1: Tạo một struct có tên PhanSo bao gồm 2 thành phần là tử số (tu) và mẫu số (mau). Tử số và mẫu số đều có kiểu dữ liệu là integer

```
struct PhanSo {
    int tu, mau;
};
```

- Bước 2: Tạo ra một hàm dùng để nhập phân số thứ nhất có tên NhapPhanSo, kiểu dữ liệu hàm trả về là void và tham số truyền vào là biến a có kiểu dữ liệu là struct PhanSo. Trong hàm tiến hành việc nhập phân số và kiểm tra xem phân số đó có hợp lệ thôi bằng cách cho một lệnh while để kiểm tra mẫu số có bằng 0. Nếu có thì nhập lại cho đến khi nhập vào một mẫu số khác 0. Sau đó kiểm tra dấu của tử số và mẫu số. Nếu cả tử và mẫu đều là số âm thì đổi dấu cả hai, còn nếu chỉ có mẫu số là số âm thì cho tử là số âm còn mẫu số là số dương.

```
7 void NhapPhanSo(PhanSo &a) {
8     cout << "Nhap phan so thu nhat: ";
9     cin >> a.tu >> a.mau;
10    while (a.mau == 0) {
11        cout << "Nhap lai: ";
12        cin >> a.tu >> a.mau;
13    }
14    if (a.mau < 0 && a.tu < 0) {
15        a.tu = abs(a.tu);
16        a.mau = abs(a.mau);
17    }
18    else if (a.mau < 0 && a.tu > 0) {
19        a.tu = (-1) * a.tu;
20        a.mau = abs(a.mau);
21    }
22 }
```

- Bước 3: Tạo ra hàm dùng để nhập vào phân số thứ hai, kiểu dữ liệu trả về là struct PhanSo và không có tham số truyền vào. Trong hàm, tạo một biến b có kiểu dữ liệu struct PhanSo và có chức năng tương tự với hàm NhapPhanSo ở bước 2.

```
PhanSo Nhap() {
    PhanSo b;
    cout << "Nhap phan so thu hai: ";
    cin >> b.tu >> b.mau;
    while (b.mau == 0) {
        cout << "Nhap lai: ";
        cin >> b.tu >> b.mau;
    }
    if (b.mau < 0 && b.tu < 0) {
        b.tu = abs(b.tu);
        b.mau = abs(b.mau);
    }
    if (b.mau < 0) {
        b.tu = (-1) * b.tu;
        b.mau = abs(b.mau);
    }
    return b;
}
```

- Bước 4: Tạo tiếp một hàm khác dùng để so sánh hai phân số vừa nhập vào tên PhanSoMax có kiểu dữ liệu trả về là struct PhanSo. Truyền vào hàm là hai biến a và b có kiểu dữ liệu là struct PhanSo. Trong hàm thực hiện bước quy đồng với tử số của hai phân số và tiến hành so sánh. Nếu tử số a lớn hơn tử số b thì trả về phân số a và ngược lại.

```
PhanSo PhanSoMax(PhanSo a, PhanSo b) {  
    if (a.tu*b.mau > b.tu*a.mau) return a;  
    else return b;  
}
```

- Bước 5: Tạo thêm một hàm để xuất ra phân số lớn nhất, đặt tên là Xuat, kiểu dữ liệu là void, tham số truyền vào là kết quả của phép so sánh hai phân số có kiểu dữ liệu là struct PhanSo. Trong hàm, nếu tử chia hết mẫu thì xuất ra tử của phân số, ngược lại thì xuất cả tử lẫn mẫu của phân số.

```
void Xuat(PhanSo c) {  
    if (c.tu % c.mau == 0) cout << "Phan so lon nhat la: " << c.tu;  
    else cout << "Phan so lon nhat la: " << c.tu << "/" << c.mau;  
}
```

- Bước 6: Trong hàm main ta tạo ba biến a, b, c có cùng kiểu dữ liệu struct PhanSo. Trong đó a sẽ được gán giá trị của phân số thứ nhất nhập vào, b được gán giá trị của phân số thứ hai nhập vào và c sẽ được gán giá trị của phân số lớn nhất trong hai phân số trên. Sau đó sử dụng hàm Xuat để xuất giá trị của phân số kết quả.

```
int main()  
{  
    PhanSo a, b, c;  
    NhapPhanSo(a);  
    b = Nhap();  
    c = PhanSoMax(a, b);  
    Xuat(c);  
    return 0;  
}
```

**3. Câu hỏi 03:**

- Mô tả/mục tiêu: Viết chương trình nhập vào hai phân số. Tính tổng, hiệu, tích, thương giữa chúng và xuất kết quả.
- Các bước thực hiện/ Phương pháp thực hiện:
  - Bước 1: Tạo một struct có tên PhanSo bao gồm 2 thành phần là tử số (tu) và mẫu số (mau). Tử số và mẫu số đều có kiểu dữ liệu là integer

```
struct PhanSo {  
    int tu, mau;  
};
```

- Bước 2: Tạo hai hàm dùng để chỉnh sửa hai phân số nhập vào. Một hàm tên RutGon không có kiểu trả về (void), có tham số truyền vào là tham chiếu của biến a có dạng struct PhanSo. Bên trong hàm, tạo một biến kiểu số nguyên temp và gán giá trị của phép ước chung lớn nhất (hàm gcd của thư viện numeric) của tử và mẫu tham số truyền vào. Rồi tiến hành chia cả tử và mẫu cho temp để rút gọn phân số. Hàm còn lại là DoiDau cũng không có kiểu trả về, cũng có tham số truyền vào là tham chiếu của biến a có dạng struct PhanSo. Trong hàm sẽ kiểm tra xem phân số nhập vào có cả tử và mẫu có ở số âm không. Nếu có thì tiến hành đổi dấu, còn không thì nếu như chỉ có mẫu ở số âm thì cho tử là số âm, còn mẫu số là số dương.

```
void DoiDau(PhanSo &a) {  
    if (a.mau < 0 && a.tu < 0) {  
        a.tu = abs(a.tu);  
        a.mau = abs(a.mau);  
    }  
    else if (a.mau < 0 && a.tu > 0) {  
        a.tu = (-1) * a.tu;  
        a.mau = abs(a.mau);  
    }  
}  
  
void RutGon(PhanSo &a) {  
    int temp = gcd(a.tu, a.mau);  
    a.tu = a.tu / temp;  
    a.mau = a.mau / temp;  
}
```

- Bước 3: Tạo hai hàm dùng để lần lượt nhập vào phân số thứ nhất và phân số thứ hai. Trong đó một hàm tên NhapPhanSo không có kiểu dữ liệu trả về (void), có tham số truyền vào là tham chiếu của biến a có dạng struct PhanSo. Trong hàm cho phép nhập tử số và mẫu số và chỉ tiến hành bước nhập lại nếu như có mẫu số nhập vào bằng 0. Sau đó gọi hàm DoiDau ở đổi dấu cho cả tử và mẫu (nếu có số âm). Hàm thứ hai là Nhap có kiểu trả về là struct PhanSo, không cần tham số truyền vào và có chức năng tương tự như hàm NhapPhanSo.

```

void NhapPhanSo(PhanSo& a) {
    cout << "Nhap phan so thu nhat: ";
    cin >> a.tu >> a.mau;
    while (a.mau == 0) {
        cout << "Nhap lai: ";
        cin >> a.tu >> a.mau;
    }
    DoiDau(a);
}

PhanSo Nhap() {
    PhanSo b;
    cout << "Nhap phan so thu hai: ";
    cin >> b.tu >> b.mau;
    while (b.mau == 0) {
        cout << "Nhap lai: ";
        cin >> b.tu >> b.mau;
    }
    DoiDau(b);
    return b;
}

```

- Bước 4: Lần lượt tạo ra các hàm để thực hiện các phép toán cộng, trừ, nhân, chia cho hai phân số vừa nhập vào. Cả 4 hàm đều có kiểu dữ liệu trả về là struct PhanSo, cùng cần hai tham số truyền vào là a và b có cùng kiểu dữ liệu struct PhanSo và đều in ra kết quả của phép tính đó. Trong hàm cộng và trừ, tạo một biến kiểu struct PhanSo tên result và sử dụng quy đồng để thực hiện các phép tính, sau đó lần lượt gán giá trị phép tính cho biến result và gọi đến hàm RutGon để rút gọn kết quả. Còn trong hàm nhân và chia, kết quả sẽ là tích của tử hai phân số chia cho tích của mẫu hai phân số, đối với chia là phép nhân nghịch đảo đối với phân số b và cũng đều gọi đến hàm RutGon để rút gọn kết quả về phân số tối giản.

```

42 void Tong(PhanSo a, PhanSo b) {
43     a.tu = a.tu * b.mau + b.tu * a.mau;
44     a.mau = a.mau * b.mau;
45     RutGon(a);
46     if (a.tu % a.mau == 0) cout << "Tong hai phan so: " << a.tu << endl;
47     else cout << "Tong hai phan so: " << a.tu << "/" << a.mau << endl;
48 }
49 void Hieu(PhanSo a, PhanSo b) {
50     a.tu = a.tu * b.mau - b.tu * a.mau;
51     a.mau = a.mau * b.mau;
52     RutGon(a);
53     if (a.tu % a.mau == 0) cout << "Hieu hai phan so: " << a.tu << endl;
54     else cout << "Hieu hai phan so: " << a.tu << "/" << a.mau << endl;
55 }
56 void Tich(PhanSo a, PhanSo b) {
57     a.tu = a.tu * b.tu;
58     a.mau = a.mau * b.mau;
59     RutGon(a);
60     if (a.tu % a.mau == 0) cout << "Tich hai phan so: " << a.tu << endl;
61     else cout << "Tich hai phan so: " << a.tu << "/" << a.mau << endl;
62 }
63 void Thuong(PhanSo a, PhanSo b) {
64     a.tu = a.tu * b.mau;
65     a.mau = a.mau * b.tu;
66     RutGon(a);
67     if (a.tu % a.mau == 0) cout << "Thuong hai phan so: " << a.tu << endl;
68     else cout << "Thuong hai phan so: " << a.tu << "/" << a.mau << endl;
69 }

```

- Bước 5: Tạo hàm main và khai báo hai biến a và b có cùng kiểu dữ liệu là struct PhanSo. Gọi đến hàm NhapPhanSo và hàm Nhap để nhập giá trị phân số a và b. Cuối cùng gọi đến các hàm cộng trừ nhân chia để thực hiện phép tính và đồng thời xuất ra kết quả.

```

int main()
{
    PhanSo a, b;
    NhapPhanSo(a);
    b = Nhap();
    Tong(a, b);
    Hieu(a, b);
    Tich(a, b);
    Thuong(a, b);
    return 0;
}

```



#### 4. Câu hỏi 04:

- Mô tả/Mục tiêu: Viết chương trình nhập vào một ngày. Tìm ngày kế tiếp và xuất kết quả
- Các bước thực hiện/ Phương pháp thực hiện:
  - Bước 1: Tạo ra 1 hàm để kiểm tra xem năm được nhập vào có phải là năm nhuận hay không. Đó là hàm `isNamNhuân` có kiểu trả về là `bool`, có tham số truyền vào là giá trị của năm nhập vào. Nếu như năm đó chia hết cho 4 và không chia hết cho 100 hoặc là năm đó chia hết cho 400 thì trả về là `true` (đúng), còn nếu không là `false` (sai). Thuật toán kiểm tra năm nhuận có là do Trái Đất mất 365 ngày 5 giờ 59 phút và 16 giây để xoay quay Mặt Trời, và để khiến cho lịch chúng ta đang xài không bị lệch so với thời gian Trái Đất quay quanh Mặt Trời thì cứ đúng 4 năm Trái Đất sẽ lệch khoảng 23 giờ 57 phút và 4 giây so với lịch của chúng ta, nên phải bù thêm 1 ngày vào tháng 2 để tránh sự chênh lệch ấy.

```
bool isNamNhuân(int nam) {  
    if ((nam % 4 == 0 && nam % 100 != 0) || (nam % 400 == 0)) return true;  
    else return false;  
}
```

- Bước 2: Tạo ra một hàm để kiểm tra tháng nhập vào có tối đa bao nhiêu ngày, là hàm `NgayCuaThang`, có kiểu dữ liệu trả về là số nguyên (`int`), có tham số truyền vào là giá trị của tháng nhập vào và năm nhập vào và sẽ trả về số ngày của tháng đó. Trong hàm sẽ kiểm tra nếu tháng nhập vào có phải là tháng 31 ngày (tháng 1, 3, 5, 7, 8, 10, 12) hay là tháng có 30 ngày (tháng 4, 6, 9, 11). Còn riêng tháng 2 thì ta cần kiểm tra xem năm nhập vào có là năm nhuận không bằng gọi hàm `isNamNhuân` ở bước 1. Nếu đúng thì tháng có 29 ngày, ngược lại thì là 28 ngày.

```
int NgayCuaThang(int thang, int nam) {  
    if (thang == 1 || thang == 3 || thang == 5 || thang == 7 || thang == 8 || thang == 10 || thang == 12) return 31;  
    else if (thang == 2 && isNamNhuân(nam) == true) return 29;  
    else if (thang == 2 && isNamNhuân(nam) == false) return 28;  
    else if (thang == 4 || thang == 6 || thang == 9 || thang == 11) return 30;  
}
```

- Bước 3: Tạo ra một hàm để xuất ra được ngày kế tiếp của ngày được nhập vào, là hàm `TheNextDay`, không có kiểu trả về (`void`), cần truyền vào tham số là giá trị của ngày tháng năm được nhập vào. Trong hàm so sánh nếu ngày đó chưa phải là ngày cuối tháng thì cộng ngày đó thêm 1. Còn nếu ngày đó mà là ngày cuối tháng nhưng tháng nhập vào chưa phải là cuối năm (tháng 12) thì reset ngày về ngày 1 và tháng đó được cộng thêm 1. Còn nếu như mà là ngày 31 tháng 12 thì thực hiện cộng năm nhập vào thêm 1 và reset ngày và tháng về 1. Riêng đối với tháng 2 thì nếu là nhập vào là 29/2 và là năm nhuận thì ngày tiếp theo sẽ reset về 1 và tháng được cộng thêm 1. Và nếu như nhập vào là 28/2 của năm không nhuận thì cũng sẽ reset ngày về 1 và tháng cộng thêm 1. Cuối cùng xuất ra ngày tháng năm kế tiếp.



```

void TheNextDay(int ngay, int thang, int nam) {
    int ngay1 = ngay;
    int thang1 = thang;
    int nam1 = nam;
    if (nam > 0 && thang > 0 && thang < 13 && ngay <= NgayCuaThang(thang, nam)) {
        ngay1++;
        if (thang != 12 && ngay == NgayCuaThang(thang, nam)) {
            ngay1 = 1;
            thang1++;
        }
        else if (thang == 12 && ngay == NgayCuaThang(thang, nam)) {
            ngay1 = 1;
            nam1++;
            thang1 = 1;
        }
        else if (thang == 2) {
            if (isNamNhuan(nam) == true) {
                if (ngay == 29) {
                    ngay1 = 1;
                    thang1++;
                }
            }
            else {
                if (ngay == 28) {
                    ngay1 = 1;
                    thang1++;
                }
            }
        }
    }
    cout << "Ngày tiếp theo là: " << ngay1 << "/" << thang1 << "/" << nam1 << endl;
}

```

- Bước 4: Khởi tạo hàm main và tạo ra ba biến ngay, thang, nam cùng kiểu dữ liệu số nguyên (int) để nhập vào ngày tháng năm. Và đồng thời kiểm tra xem ngày tháng năm được nhập vào có là ngày hợp lệ không (ngày  $\geq 1$  và  $\leq 31$ ) (tháng  $\geq 1$  và  $\leq 12$ ) (năm  $> 0$ ). Nếu không thì sẽ yêu cầu nhập lại đến khi nhập đúng thì thôi. Sau đó gọi đến hàm TheNextDay để in ra ngày tháng năm kế tiếp.

```

int main()
{
    int ngay, thang, nam;
    cout << "Nhập ngày, tháng, năm: ";
    cin >> ngay >> thang >> nam;
    while (ngay < 1 || ngay > 31 || thang < 1 || thang > 12 || nam <= 0) {
        cout << "Vui lòng nhập lại ngày, tháng, năm: ";
        cin >> ngay >> thang >> nam;
    }
    TheNextDay(ngay, thang, nam);
    return 0;
}

```

## 5. Câu hỏi 05:

- Mô tả/ Mục tiêu: Viết chương trình nhập họ tên, điểm toán, điểm văn của một học sinh. Tính điểm trung bình và xuất kết quả
- Các bước thực hiện/ Phương pháp thực hiện:
  - Bước 1: Tạo ra một struct HocSinh bao gồm các thuộc tính họ tên có kiểu dữ liệu là string, điểm toán và điểm văn đều có kiểu dữ liệu là float (vì điểm có thể là số thập phân)

```

struct HocSinh {
    string HoTen;
    float diemToan;
    float diemVan;
    float DTB;
};

```



- Bước 2: Tạo ra một hàm dùng để nhập dữ liệu của học sinh bao gồm họ tên, điểm toán và điểm văn, có tên là Nhap, không có kiểu dữ liệu trả về (void) và có tham số truyền vào là tham chiếu của một biến a có kiểu dữ liệu là struct HocSinh. Đồng thời khi nhập họ tên thì sử dụng hàm getline trong thư viện string (để có thể nhập họ tên có khoảng trắng ở giữa) và cũng như kiểm tra điều kiện nhập của điểm toán và văn. Nếu điểm toán, văn < 0 hoặc > 10 thì tiến hành nhập lại.

```
void Nhap(HocSinh &a) {
    cout << "Nhap ho ten: ";
    getline(cin, a.HoTen);
    cout << "Nhap diem Toan: ";
    cin >> a.diemToan;
    while (a.diemToan < 0 || a.diemToan > 10) {
        cout << "Nhap lai diem Toan: ";
        cin >> a.diemToan;
    }
    cout << "Nhap diem Van: ";
    cin >> a.diemVan;
    while (a.diemVan < 0 || a.diemVan > 10) {
        cout << "Nhap lai diem Van: ";
        cin >> a.diemVan;
    }
}
```

- Bước 3: Tạo ra một hàm dùng để tính điểm trung bình và đồng thời xuất ra kết quả, hàm đó là hàm Xuat không có kiểu dữ liệu trả về (void), có tham số truyền vào là giá trị của một biến a có kiểu struct HocSinh. Công thức của điểm trung bình là (Toán + Văn)/2. Sau đó tiến hành xuất ra họ tên, điểm toán, điểm văn và điểm trung bình, cách nhau bởi 1 tab ('\t'). Riêng điểm trung bình thì sử dụng lệnh printf("%.1f") để in ra kết quả làm tròn đến 1 chữ số thập phân.

```
void Xuat(HocSinh a) {
    a.DTB = (a.diemToan + a.diemVan) / 2;
    cout << a.HoTen << '\t' << a.diemToan << '\t' << a.diemVan << '\t';
    printf("%.1f", a.DTB);
}
```

- Bước 4: Tạo hàm main và trong đó, khởi tạo một biến A có kiểu dữ liệu struct HocSinh. Gọi đến hàm Nhap để nhập vào họ tên, điểm toán và điểm văn của một học sinh. Sau đó gọi đến hàm Xuat để lần lượt tính điểm trung bình và xuất ra kết quả.

```
int main()
{
    HocSinh A;
    Nhap(A);
    Xuat(A);
}
```

## 6. Câu hỏi 06

- Tài nguyên: lớp PhanSo có dữ liệu tuSo, mauSo và các hàm thành phần cộng, trừ, nhân, chia, xuất, nhập, định giá trị cho phân số.
- Mô tả/mục tiêu: Viết chương trình cho phép nhập vào hai phân số, in ra kết quả các phép toán cộng, trừ, nhân, chia hai phân số kể trên.
- Các bước thực hiện/ Phương pháp thực hiện:

### Bước 1: Khai báo thư viện

Khởi tạo những thư viện cần thiết như iostream và algorithm, thư viện <algorithm> là cần thiết để tính ước chung lớn nhất (greatest common divisor viết tắt là gcd) hoặc có thể tự viết hàm tính gcd.

```
#include<iostream>
// Hàm tính GCD
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}
```

(thuật toán tính gcd bằng đệ quy và giải thuật Euclid)

### Bước 2: Khởi tạo lớp PhanSo có dữ liệu và các hàm cần thiết

Khởi tạo lớp PhanSo với các thuộc tính và hàm ở những phạm vi truy cập phù hợp. tuSo và mauSo ta sẽ để ở private và các hàm cần thiết thì ta sẽ để ở public

```
1  #include<iostream>
2  // Hàm tính GCD
3  int gcd(int a, int b){ ... }
7  // Khởi tạo lớp PhanSo
8  class PhanSo {
9  private:
10     int tuSo, mauSo;
11 public:
12     // Hàm khởi tạo
13     PhanSo(){ ... }
17     // Hàm khởi tạo có đối số
18     PhanSo(int ts, int ms){ ... }
22     // Hàm nhập xuất
23     void nhap(){ ... }
29     // Hàm rút gọn phân số
30     PhanSo rutgon(PhanSo a){ ... }
36     // Hàm xuất phân số
37     void xuat(){ ... }
42     // Các hàm thành phần cộng, trừ, nhân, chia hai phân số
43     PhanSo operator+(PhanSo p){ ... }
49     PhanSo operator-(PhanSo p){ ... }
55     PhanSo operator*(PhanSo p){ ... }
61     PhanSo operator/(PhanSo p){ ... }
67 };
68
```

Ở hàm khởi tạo (constructor) ta sẽ khai báo biến `tuSo` và `mauSo`, mặc định sẽ là 0 và 1 vì đây là một phân số và khi ta khai báo biến lớp `PhanSo` thì mặc định của phân số đó nên bằng 0, từ đó tử số bằng 0 và mẫu số phải khác 0 để có thể là một phân số hợp lệ (ta có thể đặt `mauSo` là một số nào đó tùy ý nhưng vì đã khai báo biến `mauSo` là `int` nên việc đặt `mauSo` quá cao là không cần thiết và có thể gây `integer overflow`) và vì `tuSo` và `mauSo` là những số có kiểu dữ liệu `int` nên có thể gây `integer overflow` nếu nhập biến quá lớn.

```
public:
    // Hàm khởi tạo
    PhanSo() {
        tuSo = 0;
        mauSo = 1;
    }
```

Hoặc ta có thể định nghĩa biến `tuSo` và `mauSo` ở phần `private` và không cần khai báo constructor như trên (ví dụ minh họa)

```
8 class PhanSo {
9 private:
10     int tuSo=0, mauSo=1;
11 public:
```

Tiếp đến là một hàm khởi tạo khác (constructor overloading) để truyền đối số, thuận tiện hơn cho việc khởi tạo một biến lớp `PhanSo` nhanh và thuận tiện hơn cho việc khởi tạo lớp với dữ liệu có sẵn. Ta sẽ không cho phép trường hợp `mauSo=0` nên nếu `mauSo=0` thì mặc định `tuSo=0` và `mauSo=1`

```
17 // Hàm khởi tạo có đối số
18 PhanSo(int ts, int ms) {
19     while (true) {
20         if (ms == 0) { std::cout << "mau so phai khac 0!\nHay khai bao lai\n"; break; }
21         tuSo = ts;
22         mauSo = ms;
23         break;
24     }
25 }
```

Sau đó là hàm nhập hay có thể gọi là hàm cập nhật. Hàm này sẽ cho ta thay đổi giá trị mặc định của các biến `tuSo` và `mauSo`. Trong hàm này ta sẽ chỉ cho phép nhập những giá trị `tuSo` nguyên và những giá trị `mauSo` nguyên khác 0. Vì vậy nên nếu `mauSo = 0` ta sẽ tiến hành nhập lại phân số.

```
22 // Hàm nhập
23 void nhap() {
24     std::cout << "Nhap tu so va mau so:";
25     while (true) {
26         std::cin >> tuSo >> mauSo;
27         if (mauSo != 0) break;
28         std::cout << "Mau so phai khac 0!\nHay nhap lai tu so va mau so:";
29     }
30 }
```

Hàm rút gọn phân số sẽ cần thiết để có thể rút gọn phân số. Ta thực hiện bằng cách lấy gcd của `tuSo` và `mauSo` gán cho biến `temp` và chia lại cho `temp` để được một phân số rút gọn. Hàm này có thể không có kiểu trả về (`void`) và truyền tham chiếu (hoặc khởi tạo theo những cách khác với kiểu trả về khác)

```

31 // Hàm rút gọn phân số
32 void rutgon(PhanSo &a) {
33     int temp = gcd(a.mauSo, a.tuSo);
34     a.tuSo /= temp;
35     a.mauSo /= temp;
36 }

```

ví dụ:  $tuSo = -4$ ,  $mauSo = -6$  thì  $temp = gcd(-4, -6) = 2$

⇒  $tuSo = tuSo / 2 = -4 / 2 = -2$

⇒  $mauSo = mauSo / 2 = -6 / 2 = -3$

```

31 // Hàm rút gọn phân số
32 PhanSo rutgon(PhanSo a) {
33     int temp = gcd(a.mauSo, a.tuSo);
34     a.tuSo /= temp;
35     a.mauSo /= temp;
36     return a;
37 }

```

(một cách viết hàm rút gọn khác với kiểu trả về PhanSo)

Với hàm xuất phân số ta sẽ tiến hành rút gọn phân số bằng con trỏ this (\*this) trước rồi in ra theo nguyên tắc sau.

```

37 // Hàm xuất phân số
38 void xuat() {
39     rutgon(*this);
40     if (mauSo == 1) std::cout << tuSo << "\n"; // nếu mauSo = 1 thì in ra tuSo
41     else if ((tuSo > 0 && mauSo < 0) || (tuSo < 0 && mauSo < 0)) // Nếu tuSo > 0 và mauSo < 0 hay tuSo < 0 và mauSo < 0
42         std::cout << -tuSo << "/" << -mauSo << "\n"; // in ra đối của tử số và mẫu số
43         // Ex: -6/-4 = 3/2, 4/-5 = -4/5
44     else std::cout << tuSo << "/" << mauSo << "\n"; // các trường hợp còn lại thì in tuSo và mauSo
45 }

```

Cuối cùng là khai báo những hàm toán tử +, -, \*, / có kiểu trả về là PhanSo theo nguyên tắc cộng trừ nhân chia phân số thông thường, sau đó rút gọn và trả về.

```

46 // Các hàm thành phần cộng, trừ, nhân, chia hai phân số
47 PhanSo operator+(PhanSo p) {
48     PhanSo tong; // khởi tạo biến PhanSo tong
49     tong.tuSo = tuSo * p.mauSo + mauSo * p.tuSo; // thực hiện cộng phân số bằng quy đồng và gán cho tong
50     tong.mauSo = mauSo * p.mauSo;
51     rutgon(tong); // rút gọn tong
52     return tong; // trả về
53 }
54 PhanSo operator-(PhanSo p) {
55     PhanSo hieu; // khởi tạo biến PhanSo hieu
56     hieu.tuSo = tuSo * p.mauSo - mauSo * p.tuSo; // thực hiện trừ phân số bằng quy đồng và gán cho hieu
57     hieu.mauSo = mauSo * p.mauSo;
58     rutgon(hieu); // rút gọn hieu
59     return hieu; // trả về
60 }
61 PhanSo operator*(PhanSo p) {
62     PhanSo tich; // khởi tạo biến PhanSo tich
63     tich.tuSo = tuSo * p.tuSo; // thực hiện nhân 2 phân số và gán cho tich
64     tich.mauSo = mauSo * p.mauSo;
65     rutgon(tich); // rút gọn tich
66     return tich; // trả về
67 }
68 PhanSo operator/(PhanSo p) {
69     PhanSo thuong; // khởi tạo biến PhanSo thuong
70     thuong.tuSo = tuSo * p.mauSo; // thực hiện chia 2 phân số và gán cho thuong
71     thuong.mauSo = mauSo * p.tuSo;
72     rutgon(thuong); // rút gọn thuong
73     return thuong; // trả về
74 }
75 }

```

Bước 3: Khai báo hàm main

Đầu tiên ta khai báo và nhập hai biến lớp PhanSo là ps1 và ps2 theo 2 cách:

Cách 1: dùng hàm nhập của lớp PhanSo ps1.nhap()

Cách 2: dùng hàm khởi tạo có đối số với hai biến số nguyên a và b có sẵn và đã có giá trị sau khi nhập.

```
81 int main() {
82     PhanSo ps1;
83     std::cout << "Nhap phan so 1:\n";
84     ps1.nhap(); // Cách 1
85     std::cout << "Nhap tu va mau cua phan so 2:";
86     int a, b; std::cin >> a >> b;
87     PhanSo ps2(a, b); // Cách 2
```

Tiếp đến ta sẽ xuất ps1 và ps2

```
88     std::cout << "Phan so 1: ";
89     ps1.xuat();
90     std::cout << "Phan so 2: ";
91     ps2.xuat();
```

Cuối cùng ta tính toán những phép toán cộng, trừ, nhân, chia hai phân số ps1 và ps2 và xuất ra kết quả các phép toán.

```
92     std::cout << "Tong hai phan so: ";
93     (ps1 + ps2).xuat();
94     std::cout << "Hieu hai phan so: ";
95     (ps1 - ps2).xuat();
96     std::cout << "Tich hai phan so: ";
97     (ps1 * ps2).xuat();
98     std::cout << "Thuong hai phan so: ";
99     (ps1 / ps2).xuat();
100     return 0;
101 }
```

```
Microsoft Visual Studio Debug Console
Nhap phan so 1:
Nhap tu so va mau so: 5 6
Nhap tu va mau cua phan so 2: 4 -5
Phan so 1: 5/6
Phan so 2: -4/5
Tong hai phan so: 1/30
Hieu hai phan so: 49/30
Tich hai phan so: -2/3
Thuong hai phan so: -25/24
```

(ví dụ demo)

## 7. Câu hỏi 07

- Tài nguyên: Xây dựng lớp biểu diễn khái niệm số phức với hai thành phần dữ liệu thực, ảo và các hàm thành phần xuất, nhập, định giá trị cho số phức, cộng, trừ, nhân, chia hai số phức.
- Mô tả/mục tiêu: Viết chương trình cho phép nhập vào hai số phức, in ra kết quả các phép toán cộng, trừ, nhân, chia hai số phức kể trên.
- Các bước thực hiện/ Phương pháp thực hiện:

Vì bài này khá tương tự với bài trên nên cũng sẽ gồm 3 bước chính.

Bước 1: khai báo thư viện

```
1 #include <iostream>
```

## Bước 2: Khai báo lớp sophuc có dữ liệu và các hàm cần thiết

Khởi tạo lớp sophuc với các thuộc tính và hàm ở những phạm vi truy cập phù hợp. Thuộc tính thực và ảo ta sẽ để ở private và các hàm cần thiết thì để ở public.

```

1  #include <iostream>
2  // Khởi tạo lớp sophuc
3  class sophuc {
4  private:
5      int thuc, ao;
6  public:
7      // Hàm khởi tạo
8      sophuc() { ... }
9      // Hàm truyền đối số
10     sophuc(int a, int b) { ... }
11     // Hàm nhập
12     void nhap() { ... }
13     // Hàm xuất
14     void xuat() { ... }
15     // Các hàm thành phần cộng, trừ, nhân, chia hai số phức
16     sophuc operator+(sophuc p) { ... }
17     sophuc operator-(sophuc p) { ... }
18     sophuc operator*(sophuc p) { ... }
19     sophuc operator/(sophuc p) { ... }
20 };

```

Đầu tiên là 2 hàm khởi tạo (Constructor), hàm khởi tạo mặc định và hàm truyền đối số với cấu trúc và các giá trị mặc định khá giống với bài trên. Nhưng với giá trị mặc định của số phức thì thực và ảo không cần phải khác 0 nhưng bắt buộc phải là số nguyên nên có thể gán giá trị mặc định của thuc và ao là 0, cùng với đó hàm truyền đối số cũng không cấm các trường hợp = 0 nên không loại trừ trường hợp thuc hay ao = 0.

```

6  public:
7      // Hàm khởi tạo
8      sophuc() {
9          thuc = 0;
10         ao = 0;
11     }
12     // Hàm truyền đối số
13     sophuc(int a, int b) {
14         thuc = a;
15         ao = b;
16     }

```

Tiếp đến là hàm nhập, hàm này sẽ cập nhật lại giá trị của thuc và ao và có thể tùy chỉnh cho mục đích sử dụng.

```

17 // Hàm nhập
18 void nhap() {
19     std::cin >> thuc >> ao;
20 }

```

Và tiếp theo là hàm xuất. Ta sẽ xuất thuc và ao theo đúng định dạng số phức

```

21 // Hàm xuất
22 void xuat() {
23     if (ao == 0) std::cout << thuc << "\n";
24     else if (ao < 0) std::cout << thuc << ao << "i\n";
25     else std::cout << thuc << "+" << ao << "i\n";
26 }

```

Ex: thuc=0, ao=-2 => 0-2i

thuc =-2, ao=4 => -2+4i

thuc=2, ao=0 => 2

Cuối cùng là các hàm toán tử +, -, \*, / hai số phức với các công thức tính như sau:

Ví dụ: Cho hai số phức  $A(a_1, a_2)$ ,  $B(b_1, b_2)$

- $A + B = (a_1 + b_1, a_2 + b_2)$
- $A - B = (a_1 - b_1, a_2 - b_2)$
- $A * B = (a_1 * b_1 - a_2 * b_2, a_1 * b_2 + a_2 * b_1)$
- $A / B = \left( \frac{a_1 * b_1 + a_2 * b_2}{b_1^2 + b_2^2}, \frac{b_1 * a_2 - a_1 * b_2}{b_1^2 + b_2^2} \right)$

Với cách tính số phức như trên ta thực hiện các phép toán bằng cách khởi tạo các hàm với kiểu trả về là `sophuc` và khởi tạo `sophuc temp`, tính toán các phép toán sau đó gán cho `temp` và trả về `temp`. Ta có thể sử dụng con trỏ `this` để dễ phân biệt hơn.

```

27 // Các hàm thành phần cộng, trừ, nhân, chia hai số phức
28 sophuc operator+(sophuc p) {
29     sophuc temp;
30     temp.thuc = p.thuc + this->thuc;
31     temp.ao = p.ao + this->ao;
32     return temp;
33 }
34 sophuc operator-(sophuc p) {
35     sophuc temp;
36     temp.thuc = this->thuc - p.thuc;
37     temp.ao = this->ao - p.ao;
38     return temp;
39 }
40 sophuc operator*(sophuc p) {
41     sophuc temp;
42     temp.thuc = this->thuc * p.thuc - this->ao * p.ao;
43     temp.ao = this->thuc * p.ao + this->ao * p.thuc;
44     return temp;
45 }
46 sophuc operator/(sophuc p) {
47     sophuc temp;
48     temp.thuc = (this->thuc * p.thuc + this->ao * p.ao) / (p.ao * p.ao + p.thuc * p.thuc);
49     temp.ao = (p.thuc * this->ao - p.ao * this->thuc) / (p.ao * p.ao + p.thuc * p.thuc);
50     return temp;
51 }
52 };

```

### Bước 3: Khai báo hàm main

Đầu tiên ta khai báo và nhập hai biến lớp `sophuc` là `p1` và `p2` theo 2 cách:

Cách 1: dùng hàm nhập của lớp `sophuc` `p1.nhap()`

Cách 2: dùng hàm khởi tạo có đối số với hai biến số nguyên `a` và `b` có sẵn và đã có giá trị sau khi nhập.

```

50 int main(){
51     sophuc p1;
52     std::cout << "Nhập phần thực và phần ảo của số phức 1:";
53     p1.nhap();
54     std::cout << "Nhập phần thực và phần ảo của số phức 2:";
55     int a,b; std::cin >> a >> b;
56     sophuc p2(a,b);

```

Tiếp đến ta xuất `p1` và `p2`

```

61     std::cout << "Số phức 1: ";
62     p1.xuat();
63     std::cout << "Số phức 2: ";
64     p2.xuat();

```



Cuối cùng ta tính toán những phép toán cộng, trừ, nhân, chia hai phân số p1 và p2 và xuất ra kết quả các phép toán.

```

65     std::cout << "Phép cộng: ";
66     (p1 + p2).xuat();
67     std::cout << "Phép trừ: ";
68     (p1 - p2).xuat();
69     std::cout << "Phép nhân: ";
70     (p1 * p2).xuat();
71     std::cout << "Phép chia: ";
72     (p1 / p2).xuat();
73     return 0;
74 }

```

```

Microsoft Visual Studio Debug Console
Nhap phan thuc va ao cua so phuc 1:5 6
Nhap phan thu va ao cua so phuc 2:3 -2
So phuc 1: 5+6i
So phuc 2: 3-2i
Phép cộng: 8+4i
Phép trừ: 2+8i
Phép nhân: 27+8i
Phép chia: 0+2i

```

(ví dụ demo)

## 8. Câu hỏi 08

- Tài nguyên: Xây dựng lớp Candidate (Thí sinh) gồm các thuộc tính: mã, tên, ngày tháng năm sinh, điểm thi Toán, Văn, Anh và các phương thức cần thiết.
- Mô tả/mục tiêu: Xây dựng lớp TestCandidate để kiểm tra lớp trên:
  - Nhập vào n thí sinh (n do người dùng nhập).
  - In ra thông tin về các thí sinh có tổng điểm lớn hơn 15.
- Các bước thực hiện/ Phương pháp thực hiện:

Bước 1: khai báo thư viện và namespace

Khởi tạo những thư viện cần thiết như <iostream> và <string>, thư viện <string> được thêm vào để có thể lưu trữ các biến có nhiều ký tự. Tiếp theo có thể khai báo namespace standard (using namespace std) để có thể sử dụng những câu lệnh như std::cin, std::cout, ... trở nên ngắn gọn hơn.

```

1 #include<iostream>
2 #include<string>
3 using namespace std;

```

Bước 2: Khai báo lớp Candidate cùng các thuộc tính và các phương thức cần thiết.

Các thuộc tính của Candidate gồm MSSV (int id), Họ tên (string name), ngày sinh (string dob) và điểm 3 môn toán, văn, anh (float math, literature, english) sẽ được để ở private. Còn các phương thức sẽ được định vị ở public bao gồm hàm nhập (input()), hàm xuất (output()) và hàm lấy tổng điểm 3 môn toán, văn và anh (getTotalScore()).

```

4 // Khởi tạo lớp Candidate
5 class Candidate {
6 private:
7     int id; // MSSV, id
8     string name; // Họ và tên
9     string dob; // Ngày sinh
10    float math, literature, english; // điểm 3 môn toán, văn, anh
11 public:
12    // Hàm nhập
13    void input() { ... }
14    // Hàm xuất
15    void output() { ... }
16    // Hàm lấy tổng điểm 3 môn
17    float getTotalScore() { ... }
18 };

```

Về hàm nhập ta sẽ hướng dẫn cách nhập từng thuộc tính và tiến hành cập nhật các thuộc tính của lớp. Đương nhiên ta có thể thêm những thuật toán để kiểm tra tính đúng đắn của dữ liệu nhập vào như điểm phải bé hơn hoặc bằng 10 và lớn hơn hoặc bằng 0, định dạng tên được chia ra thành Họ, Tên, ... Nhưng không quá cần thiết trong bài tập này nên có thể không thêm vào.

```

11 public:
12     // Hàm nhập
13     void input() {
14         cout << "Enter candidate id: ";
15         cin >> id;
16         cout << "Enter candidate name: ";
17         cin.ignore();
18         getline(cin, name);
19         cout << "Enter date of birth (dd/mm/yyyy): ";
20         getline(cin, dob);
21         cout << "Enter math score: ";
22         cin >> math;
23         cout << "Enter literature score: ";
24         cin >> literature;
25         cout << "Enter english score: ";
26         cin >> english;
27     }

```

Về hàm xuất ta sẽ in lần lượt các thuộc tính của lớp Candidate

```

28     // Hàm xuất
29     void output() {
30         cout << "Candidate id: " << id << endl;
31         cout << "Candidate name: " << name << endl;
32         cout << "Date of birth: " << dob << endl;
33         cout << "Math score: " << math << endl;
34         cout << "Literature score: " << literature << endl;
35         cout << "English score: " << english << endl;
36     }

```

Cuối cùng là hàm lấy tổng 3 điểm toán, văn, anh với kiểu trả về float

```

37     // Hàm lấy tổng điểm 3 môn
38     float getTotalScore() {
39         return math + literature + english;
40     }
41 }

```

Bước 3: Khởi tạo lớp TestCandidate

Ta sẽ khởi tạo lớp TestCandidate với một hàm duy nhất là hàm static void run() có vai trò nhập n số thí sinh và kiểm tra tổng số điểm 3 môn của n số thí sinh đó, nếu lớn hơn 15 ta sẽ in ra thông tin của thí sinh.

```

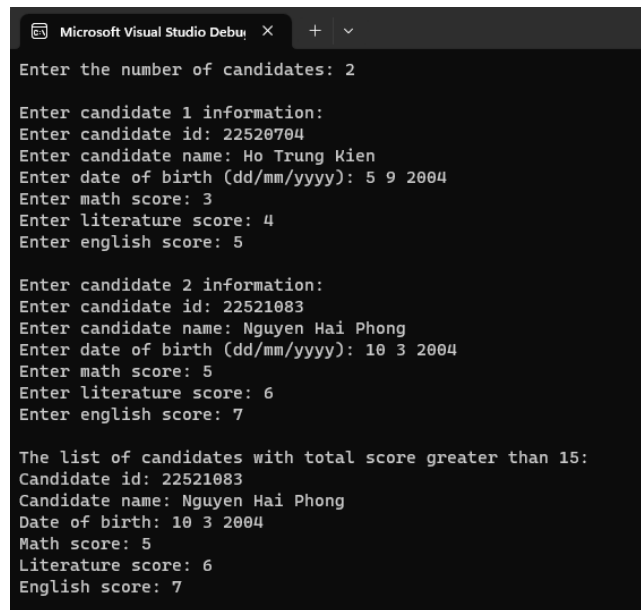
42 class TestCandidate {
43 public:
44     static void run(){
45         int n;
46         cout << "Enter the number of candidates: ";
47         cin >> n; // Nhập số ứng viên Candidate cần xét
48         Candidate* list = new Candidate[n]; // Khởi tạo mảng động n phần tử kiểu lớp Candidate
49         // Nhập thông tin của n phần tử kiểu lớp Candidate
50         for (int i = 0; i < n; i++) {
51             cout << "\nEnter candidate " << i + 1 << " information:\n";
52             list[i].input();
53         }
54         cout << "\nThe list of candidates with total score greater than 15:\n";
55         // Xuất thông tin của các Candidate có tổng số điểm 3 môn lớn hơn 15
56         for (int i = 0; i < n; i++) {
57             if (list[i].getTotalScore() > 15) {
58                 list[i].output();
59                 cout << endl;
60             }
61         }
62     }
63 };

```

## Bước 4: Khai báo hàm main

Trong hàm main ta sẽ gọi tới hàm run() của lớp TestCandidate để có thể khởi động hàm run() của lớp TestCandidate và kết thúc chương trình sau khi đã chạy chương trình hoàn tất.

```
64 int main() {  
65     TestCandidate::run();  
66     return 0;  
67 }
```



The screenshot shows the Microsoft Visual Studio Debug console with the following output:

```
Enter the number of candidates: 2  
  
Enter candidate 1 information:  
Enter candidate id: 22520704  
Enter candidate name: Ho Trung Kien  
Enter date of birth (dd/mm/yyyy): 5 9 2004  
Enter math score: 3  
Enter literature score: 4  
Enter english score: 5  
  
Enter candidate 2 information:  
Enter candidate id: 22521083  
Enter candidate name: Nguyen Hai Phong  
Enter date of birth (dd/mm/yyyy): 10 3 2004  
Enter math score: 5  
Enter literature score: 6  
Enter english score: 7  
  
The list of candidates with total score greater than 15:  
Candidate id: 22521083  
Candidate name: Nguyen Hai Phong  
Date of birth: 10 3 2004  
Math score: 5  
Literature score: 6  
English score: 7
```

(ví dụ demo)