ĐẠI 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 1

GVHD: Nguyễn Ngọc Quí

Sinh viên thực hiện: Huỳnh Nhật Minh - 24521056

Tp. Hồ Chí Minh, 03/2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....., ngày…….....tháng……năm 2025

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

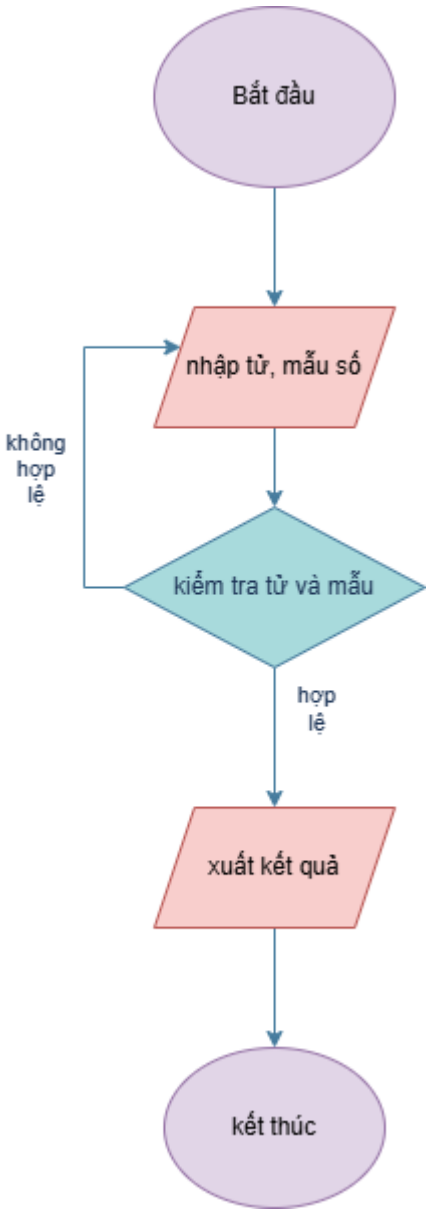
DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

NỘI DUNG BÀI LÀM

Bài tập 1: Viết chương trình nhập vào một phân số, rút gọn phân số và xuất kết quả.

Nội dung flowchart của chương trình rút gọn phân số như hình 1.



1.Hình 1

Mô tả đầu vào, đầu ra và hướng giải quyết của chương trình rút gọn phân số:

- Đầu vào: tử số và mẫu số.
- Đầu ra: phân số đã được rút gọn.
- Hướng giải quyết: tìm ước chung lớn nhất của tử số và mẫu số sau đó chia tử và mẫu cho ước chung lớn nhất này để tìm được phân số rút gọn.

Nội dung code của chương trình rút gọn phân số:

```
#include <iostream>

#include <cmath>

using namespace std;


//Khai báo struct phân số

struct phanSo

{

   int tuSo, mauSo;

};


/*Hàm nhập phân số :

- Đầu vào: nhập từ bàn phím tử số và mẫu số, nếu mẫu số = 0 thì nhập lại.

- Đầu ra: giá trị vừa nhập, dùng biến tham chiếu để lưu giá trị sau khi nhập.*/

void nhap(phanSo& ps) {

   do {

      cout << "nhap tu :";

      cin >> ps.tuSo;

      cout << "nhap mau : ";

      cin >> ps.mauSo;

                  if (ps.mauSo == 0) {

                        cout << "Phan so khong hop le! Mau so phai khac 0. Moi nhap lai."
<< endl;

                  }

   } while (ps.mauSo == 0);
```

```
}


/*Hàm tìm ước chung lớn nhất của tử và mẫu :
- Đầu vào: giá trị của biến có dạng phanSo.
- Đầu ra: trả về ước chung lớn nhất.
Sử dụng thuật toán Euclid.*/
int UCLN(phanSo ps) {
   int tu = abs(ps.tuSo), mau = abs(ps.mauSo);
   while (tu != mau) {
      if (tu > mau) {
         tu -= mau;
      }
      else {
         mau -= tu;
      }
   }
   return tu;
}


/*Hàm rút gọn phân số:
Đầu vào: giá trị của biến có dạng phanSo.
Đầu ra: in ra phân số đã được rút gọn theo đúng định dạng.
Chia cả tử và mẫu cho ước chung lớn nhất để tìm phân số dạng rút gọn*/
void rutgon(phanSo ps) {
   phanSo rutgon;
```

```
        if (ps.tuSo == 0) {
                cout << 0;
        }
    else {
        int ucln = UCLN(ps);
        rutgon.mauSo = ps.mauSo / ucln;
        if (rutgon.mauSo == 1) {
        rutgon.tuSo = ps.tuSo / ucln;
        cout << rutgon.tuSo;
        }
        else if (rutgon.mauSo == -1) {
        rutgon.tuSo = ps.tuSo / ucln;
        cout << -rutgon.tuSo;
        }
        else {
        rutgon.tuSo = ps.tuSo / ucln;
            if (rutgon.tuSo < 0 || rutgon.mauSo < 0) {
                if (rutgon.tuSo < 0 && rutgon.mauSo < 0) {
                cout << abs(rutgon.tuSo) << '/' << abs(rutgon.mauSo);
                }
                else {
                cout << '-' << abs(rutgon.tuSo) << '/' << abs(rutgon.mauSo);
                }
            }
            else {
```

```
            cout << rutgon.tuSo << '/' << rutgon.mauSo;
            }
        }
    }
}

int main()
{
    phanSo ps;
    nhap(ps);
            cout << "Phan so sau khi rut gon la: ";
    rutgon(ps);
    return 0;
}
```

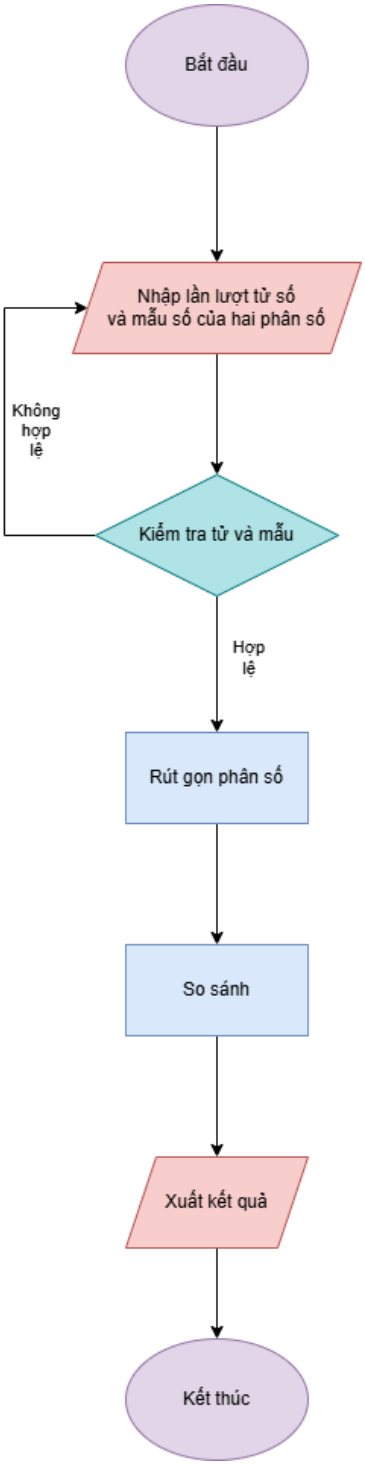
1.Bảng 1

Nội dung kiểm thử của chương trình rút gọn phân số:

- Kiểm thử 1: cho mẫu số = 0, cho tử số = 1, nhập lại.
- Kiểm thử 2: cho mẫu số = 99, cho tử số = 99, kết quả phân số được rút gọn là 1
- Kiểm thử 3: cho mẫu số = 1, cho tử số = 0, kết quả phân số được rút gọn là 0
- Kiểm thử 4: cho mẫu số = 25, cho tử số = 100, kết quả phân số được rút gọn là 4
- Kiểm thử 5: cho mẫu số = 50, cho tử số = 10, kết quả phân số được rút gọn là 1/5
- Kiểm thử 6: cho mẫu số = -45, cho tử số = 9, kết quả phân số được rút gọn là -1/5
- Kiểm thử 7: cho mẫu số = 60, cho tử số = -6, kết quả phân số được rút gọn là -1/10

Bài tập 2: Viết chương trình nhập vào hai phân số, tìm phân số lớn nhất và xuất kết quả.

Nội dung flowchart của chương trình rút gọn phân số như hình 2.



2.Hình 2

Mô tả đầu vào, đầu ra và hướng giải quyết của chương trình tìm phân số lớn nhất:

- Đầu vào: tử số và mẫu số của hai phân số.
- Đầu ra: phân số lớn nhất.

- Hướng giải quyết: rút gọn cả hai phân số sau đó tiến hành quy đồng rồi so sánh giá trị của tử số hai phân số rồi xuất kết quả.

Nội dung code của chương trình tìm phân số lớn nhất:

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


//Khai báo struct phân số
struct phanSo
{
    int tuSo, mauSo;
};


/*Hàm nhập phân số :
- Đầu vào: nhập từ bàn phím 2 phân số gồm tử số và mẫu số, nếu mẫu số = 0 thì nhập lại.
- Đầu ra: lưu giá trị vừa nhập vào list ps.*/
void nhap(phanSo ps[]) {
    for (int i = 0; i < 2; i++) {
        cout << "nhap phan so thu " << i + 1 << ": " << endl;
        do {
            cout << "nhap tu :";
            cin >> ps[i].tuSo;
            cout << "nhap mau : ";
            cin >> ps[i].mauSo;
            if (ps[i].mauSo == 0) {
                cout << "Phan so khong hop le! Mau so phai khac 0. Moi nhap lai." << endl;
            }
        } while (ps[i].mauSo == 0);
```

```
    }
}


/*Hàm tìm ước chung lớn nhất của tử và mẫu :
- Đầu vào: giá trị của biến có dạng phanSo.
- Đầu ra: trả về ước chung lớn nhất.
Sử dụng thuật toán Euclid.*/
int UCLN(const phanSo& ps) {
    int tu = abs(ps.tuSo), mau = abs(ps.mauSo);
    while (tu != mau) {
        if (tu > mau) {
            tu -= mau;
        }
        else {
            mau -= tu;
        }
    }
    return tu;
}


/*Hàm tìm bội chung nhỏ nhất để quy đồng phân số dựa vào mẫu số:
Đầu vào: giá trị của 2 mẫu số.
Đầu ra: trả về bội chung nhỏ nhất.
Chia cả tử và mẫu cho ước chung lớn nhất để tìm phân số dạng rút gọn*/
int BCNN(int a, int b) {
```

```
    int abd = a, bbd = b;
    while (a != b) {
        if (a > b) {
            if (a % b == 0) {
                return a;
                break;
            }
            else {
                a += abd;
            }
        }
        else {
            if (b % a == 0) {
                return b;
                break;
            }
            else {
                b += bbd;
            }
        }
    }
}
```

/*Hàm rút gọn phân số và trả về giá trị của phân số đó để so sánh:

Đầu vào: giá trị của biến có dạng phanSo.

```
Đầu ra: trả về phân số tối giản nhất.
Chia cả tử và mẫu cho ước chung lớn nhất để tìm phân số dạng rút gọn*/
phanSo rutGon(const phanSo& ps) {
    phanSo rutgon;
    if (ps.tuSo == 0) {
        rutgon.tuSo = 0;
        rutgon.mauSo = 1;
    }
    else {
        int ucln = UCLN(ps);
        rutgon.mauSo = ps.mauSo / ucln;
        rutgon.tuSo = ps.tuSo / ucln;
    }
    return rutgon;
}

/*Hàm quy đồng 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: trả về 2 phân số đã quy đồng.
Quy đồng bằng bội chung nhỏ nhất, dùng biến tham chiếu để lưu giá trị trả về*/
void quyDong(phanSo& ps1, phanSo& ps2) {
    int bcnn = BCNN(ps1.mauSo, ps2.mauSo);
    if (ps1.mauSo != ps2.mauSo) {
        if (ps1.mauSo < ps2.mauSo) {
            ps1.mauSo = bcnn;
```

```
            ps1.tuSo *= (bcnn / ps1.mauSo);
        }
        else {
            ps2.mauSo = bcnn;
            ps2.tuSo *= (bcnn / ps2.mauSo);
        }
    }
}
/*Hàm in kết quả:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: in ra màn hình kết quả dúng định dạng.*/
void inKQ(const phanSo& ps) {
    if (ps.tuSo == 0) {
        cout << 0 << endl;
    }
    else if (ps.mauSo ==1) {
        cout << ps.tuSo << endl;
    }
    else if (ps.tuSo < 0 && ps.mauSo < 0) {
        cout << -ps.tuSo << '/' << -ps.mauSo << endl;
    }
    else if (ps.tuSo > 0 && ps.mauSo > 0) {
        cout << ps.tuSo << '/' << ps.mauSo << endl;
    }
    else {
```

```
        cout << -abs(ps.tuSo) << '/' << abs(ps.mauSo) << endl;
    }
}


int main()
{
    phanSo ps[2], rg[2];
    nhap(ps);
    rg[0] = rutGon(ps[0]);
    rg[1] = rutGon(ps[1]);
    cout << "phan so lon nhat la: ";
    if (rg[0].tuSo == 0 || rg[1].tuSo == 0) {
        if ((rg[0].tuSo * rg[0].mauSo) >= (rg[1].tuSo * rg[1].mauSo)) {
            inKQ(ps[0]);
        }
        else {
            inKQ(ps[1]);
        }
    }
    else {
        quyDong(rg[0], rg[1]);
        if (rg[0].tuSo - rg[1].tuSo >= 0) {
            inKQ(ps[0]);
        }
        else {
```

```
            inKQ(ps[1]);
        }
    }
    return 0;
}
```

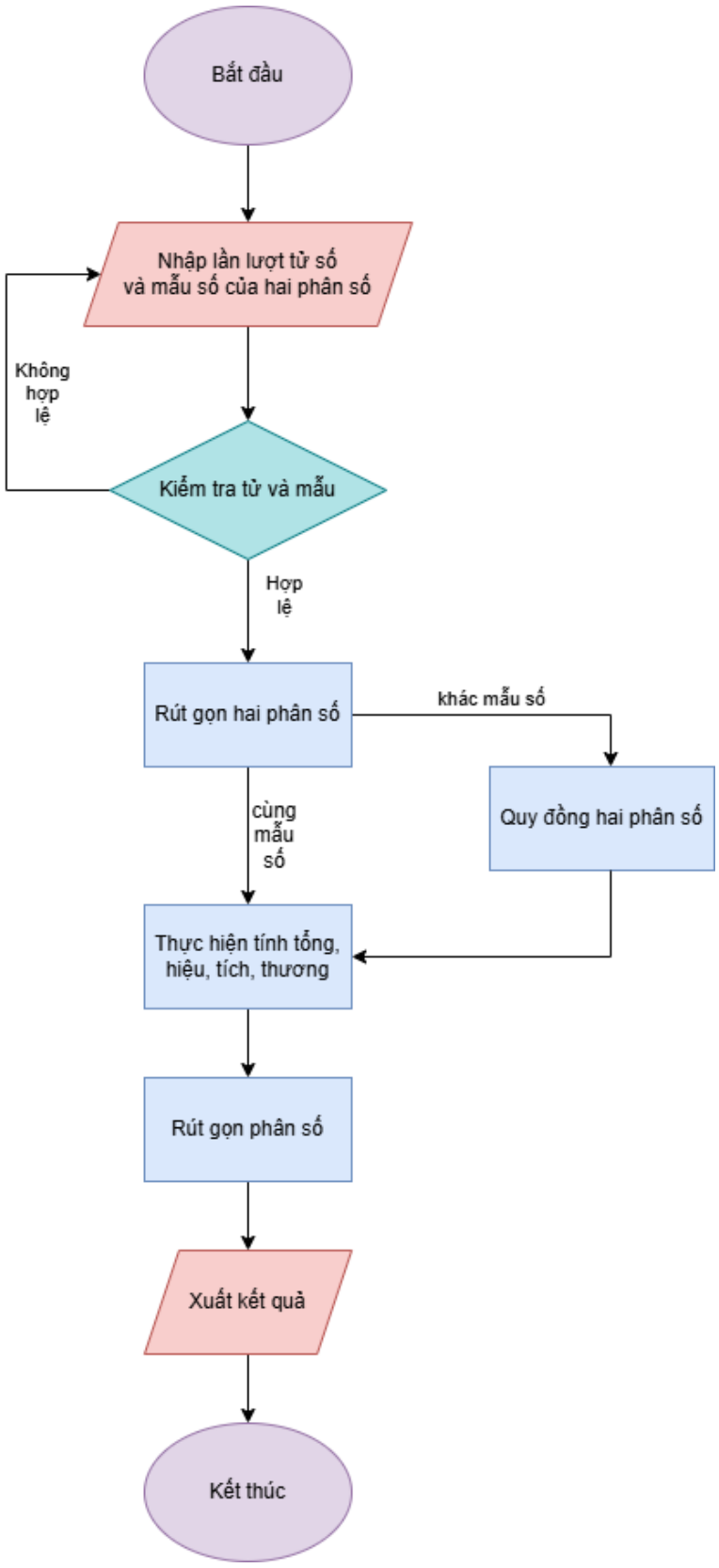
2.Bảng 2

Nội dung kiểm thử của chương trình tìm phân số lớn nhất:

- Kiểm thử 1: hai phân số lần lượt là: 5/7, 3/7. Kết quả phân số lớn nhất là: 5/7.
- Kiểm thử 2: hai phân số lần lượt là: 5/9, 7/11. Kết quả phân số lớn nhất là: 7/11.
- Kiểm thử 3: hai phân số lần lượt là: 5/-7, 5/-8. Kết quả phân số lớn nhất là: -5/7.
- Kiểm thử 4: hai phân số lần lượt là: 3/5, 3/5. Kết quả phân số lớn nhất là: 3/5.
- Kiểm thử 5: hai phân số lần lượt là: 0/3, 0/4. Kết quả phân số lớn nhất là: 0.
- Kiểm thử 6: hai phân số lần lượt là: 5/7, 0/7. Kết quả phân số lớn nhất là: 5/7.
- Kiểm thử 7 hai phân số lần lượt là: -5/7, 0/7. Kết quả phân số lớn nhất là: 0.
- Kiểm thử 8 hai phân số lần lượt là: -5/0, 1/1. Nhập lại phân số thứ nhất.
- Kiểm thử 9 hai phân số lần lượt là: -5/7, 9/0. Nhập lại phân số thứ hai.

Bài tập 3: Viết chương trình nhập vào hai phân số. Tính tổng, hiệu, tích, thương giữa chúng và xuất kết quả.

Nội dung flowchart của chương trình tính toán phân số như hình 3.



3.Hình 3

Mô tả đầu vào, đầu ra và hướng giải quyết của chương tính toán phân số:

- Đầu vào: tử số và mẫu số của hai phân số.
- Đầu ra: tổng, hiệu, tích, thương của 2 phân số.

- Hướng giải quyết: rút gọn cả hai phân số sau đó tiến hành quy đồng rồi tính toán tổng, hiệu, tích, thương, cuối cùng xuất kết quả tối giản.

Nội dung code của chương trình tính toán phân số:

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


//Khai báo struct phân số
struct phanSo
{
    int tuSo, mauSo;
};


/*Hàm nhập phân số :
- Đầu vào: nhập từ bàn phím 2 phân số gồm tử số và mẫu số, nếu mẫu số = 0 thì nhập lại.
- Đầu ra: lưu giá trị vừa nhập vào list ps.*/
void nhap(phanSo ps[]) {
    for (int i = 0; i < 2; i++) {
        cout << "nhap phan so thu " << i + 1 << ": " << endl;
        do {
            cout << "nhap tu :";
            cin >> ps[i].tuSo;
            cout << "nhap mau : ";
            cin >> ps[i].mauSo;
            if (ps[i].mauSo == 0) {
                cout << "Phan so khong hop le! Mau so phai khac 0. Moi nhap lai." << endl;
            }
        } while (ps[i].mauSo == 0);
```

```
    }
}


/*Hàm tìm ước chung lớn nhất của tử và mẫu :
- Đầu vào: giá trị của biến có dạng phanSo.
- Đầu ra: trả về ước chung lớn nhất.
Sử dụng thuật toán Euclid.*/
int UCLN(const phanSo& ps) {
    int tu = abs(ps.tuSo), mau = abs(ps.mauSo);
    while (tu != mau) {
        if (tu > mau) {
            tu -= mau;
        }
        else {
            mau -= tu;
        }
    }
    return tu;
}


/*Hàm tìm bội chung nhỏ nhất để quy đồng phân số dựa vào mẫu số:
Đầu vào: giá trị của 2 mẫu số.
Đầu ra: trả về bội chung nhỏ nhất.
Chia cả tử và mẫu cho ước chung lớn nhất để tìm phân số dạng rút gọn*/
int BCNN(int a, int b) {
```

```
    int abd = a, bbd = b;
    while (a != b) {
        if (a > b) {
            if (a % b == 0) {
                return a;
                break;
            }
            else {
                a += abd;
            }
        }
        else {
            if (b % a == 0) {
                return b;
                break;
            }
            else {
                b += bbd;
            }
        }
    }
}
```

/*Hàm rút gọn phân số và trả về giá trị của phân số đó để so sánh:

Đầu vào: giá trị của biến có dạng phanSo.

```
Đầu ra: trả về phân số tối giản nhất.
Chia cả tử và mẫu cho ước chung lớn nhất để tìm phân số dạng rút gọn*/
phanSo rutGon(phanSo ps) {
    phanSo rutgon;
    if (ps.tuSo == 0) {
        rutgon.tuSo = 0;
        rutgon.mauSo = 1;
    }
    else {
        int ucln = UCLN(ps);
        rutgon.mauSo = ps.mauSo / ucln;
        rutgon.tuSo = ps.tuSo / ucln;
    }
    return rutgon;
}

/*Hàm quy đồng 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: trả về 2 phân số đã quy đồng.
Quy đồng bằng bội chung nhỏ nhất, dùng biến tham chiếu để lưu giá trị trả về*/
void quyDong(phanSo& ps1, phanSo& ps2) {
    int bcnn = BCNN(ps1.mauSo, ps2.mauSo);
    ps1.tuSo *= (bcnn / ps1.mauSo);
    ps1.mauSo = bcnn;
    ps2.tuSo *= (bcnn / ps2.mauSo);
```

```
    ps2.mauSo = bcnn;
}
/*Hàm in kết quả:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: in ra màn hình kết quả dúng định dạng.*/
void inKQ(const phanSo& ps) {
    if (ps.tuSo == 0) {
        cout << 0 << endl;
    }
    else if (ps.mauSo == 1) {
        cout << ps.tuSo << endl;
    }
    else if (ps.tuSo < 0 && ps.mauSo < 0) {
        cout << -ps.tuSo << '/' << -ps.mauSo << endl;
    }
    else if (ps.tuSo > 0 && ps.mauSo > 0) {
        cout << ps.tuSo << '/' << ps.mauSo << endl;
    }
    else {
        cout << -abs(ps.tuSo) << '/' << abs(ps.mauSo) << endl;
    }
}

/*Hàm tính tổng 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
```

```
Đầu ra: trả về giá trị tổng của 2 phân số.*/
phanSo tinhTong(phanSo ps1, phanSo  ps2) {
   phanSo kq;
   if (ps1.mauSo != ps2.mauSo) {
      quyDong(ps1, ps2);
   }
   kq.mauSo = ps1.mauSo;
   kq.tuSo = ps1.tuSo + ps2.tuSo;
   return kq;

}

/*Hàm tính hiệu 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: trả về giá trị hiệu của 2 phân số.*/
phanSo tinhHieu(phanSo ps1, phanSo  ps2) {
   phanSo kq;
   if (ps1.mauSo != ps2.mauSo) {
      quyDong(ps1, ps2);
   }
   kq.mauSo = ps1.mauSo;
   kq.tuSo = ps1.tuSo - ps2.tuSo;
   return kq;

}
```

```
/*Hàm tính tích 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: trả về giá trị tích của 2 phân số.*/
phanSo tinhTich(phanSo ps1, phanSo ps2) {
    phanSo kq;
    kq.tuSo = ps1.tuSo * ps2.tuSo;
    kq.mauSo = ps1.mauSo * ps2.mauSo;
    return kq;
}

/*Hàm tính thương 2 phân số:
Đầu vào: giá trị của 2 biến có dạng phanSo.
Đầu ra: trả về giá trị thương của 2 phân số.*/
phanSo tinhThuong(phanSo ps1, phanSo ps2) {
    phanSo kq;
    kq.tuSo = ps1.tuSo * ps2.mauSo;
    kq.mauSo = ps1.mauSo * ps2.tuSo;
    return kq;
}

int main()
{
    phanSo ps[2], rg[2];
    nhap(ps);
```

```
    rg[0] = rutGon(ps[0]);
    rg[1] = rutGon(ps[1]);
    phanSo tong = rutGon(tinhTong(rg[0], rg[1])),
        hieu = rutGon(tinhHieu(rg[0], rg[1])),
        tich = rutGon(tinhTich(rg[0], rg[1]));
    cout << "tong 2 phan so la: ";
    inKQ(tong);
    cout << "hieu 2 phan so la: ";
    inKQ(hieu);
    cout << "tich 2 phan so la: ";
    inKQ(tich);
    cout << "thuong 2 phan so la: ";
    if (rg[1].tuSo == 0) {
        cout << "khong the chia cho 0!" << endl;
    }
    else {
        inKQ(rutGon(tinhThuong(rg[0], rg[1])));
    }
    return 0;
}
```

3.Bảng 3

Nội dung kiểm thử của chương trình tính toán phân số:

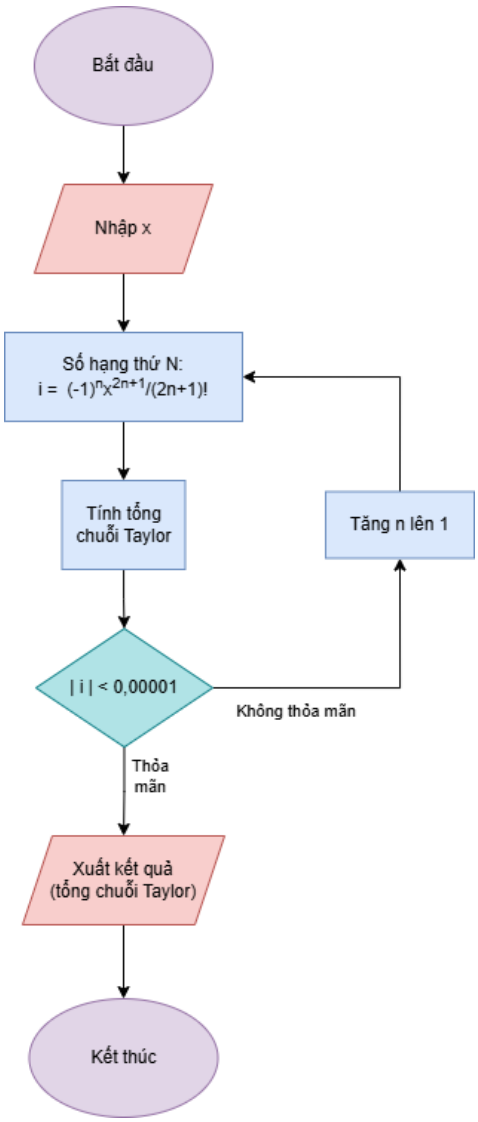
- Kiểm thử 1: hai phân số lần lượt là: 1/2, 3/4. Kết quả: tổng: 5/4, hiệu: -1/4, tích: 3/8, thương: 2/3.
- Kiểm thử 2: hai phân số lần lượt là: -4/5, -3/2. Kết quả: tổng: -23/10, hiệu: 7/10, tích: 6/5, thương: 8/15.

- Kiểm thử 3: hai phân số lần lượt là: 0/1, 2/5. Kết quả: tổng: 2/5, hiệu: -2/5, tích: 0, thương: 0.
- Kiểm thử 4: hai phân số lần lượt là: 0, 0/5. Kết quả: tổng: 0, hiệu: 0, tích: 0, thương: 0.
- Kiểm thử 5: hai phân số lần lượt là: 5/1 ,6/1. Kết quả: tổng: 11, hiệu: -1, tích: 30, thương: 5/6.
- Kiểm thử 6: hai phân số lần lượt là: 1/0, 2/5. Nhập lại phân số thứ nhất.
- Kiểm thử 7: hai phân số lần lượt là: 1/2, 2/0. Nhập lại phân số thứ hai.

Bài tập 4: Lập chương trình tính Sin(x) với độ chính xác 0.00001 theo công thức:

$$Sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

Nội dung flowchart của chương trình tính Sin(x) như hình 4.



4. Hình 4

Mô tả đầu vào, đầu ra và hướng giải quyết của chương trình tính Sin(x):

- Đầu vào: một số x.
- Đầu ra: Sin(x) với độ chính xác 0.00001.
- Hướng giải quyết: thế x vào phương trình, tiếp tục vừa cộng vừa tăng giá trị n, kết hợp so sánh số hạng cuối đến khi nào thỏa mãn điều kiện : giá trị tuyệt đối của số hạng đó < 0,00001, lập tức kết thúc và trả về kết quả.

Nội dung code của chương trình tính Sin(x):

```
#include <iostream>

#include <cmath>

#include <iomanip>
```

```
using namespace std;


/*Hàm nhập x:
Đầu vào: giá trị của biến x.
Đầu ra: lưu giá trị của biến x thông qua biến tham chiếu.*/
void Nhap(double& x) {
	cout << "Nhap x: ";
	cin >> x;
}


/*Hàm tính sin(x):
Đầu vào: giá trị của biến x.
Đầu ra: giá trị của sin(x).
Dùng biểu thức Taylor , kiểm tra điều kiện của số hạng cuối cùng
nếu bé hơn độ chính xác tối thiểu sẽ kết thúc và in kết quả */
void TinhSin(double x) {
	double k = 0;
	double i = 1;
	double mau = 1;
	double tu = x;
	int n = 0;
	while (abs(i)>= 0.00001) {
		i =  tu / mau;
		k += i;
		tu *= -x * x;
```

```
            n++;
            mau = mau * (2 * n) * (2 * n + 1);
        }
        if (k<=0.0001) {
            cout <<  "Sin(" << x << ") = " << fixed << setprecision(5) << k << endl;
        }
        else {
            cout << "Sin(" << x << ") = " << k << endl;
        }
}

int main() {
        double x;
        Nhap(x);
        TinhSin(x);
}
```

4.Bảng 4

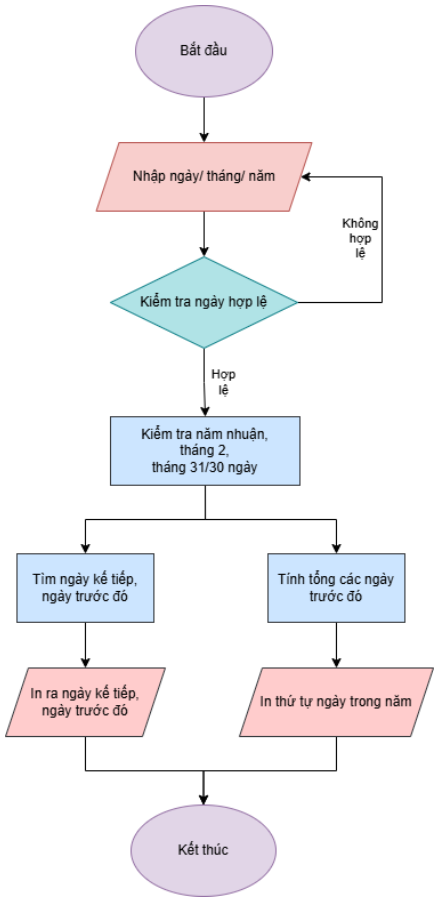
Nội dung kiểm thử của chương trình tính Sin(x):

- Kiểm thử 1: x = 0, kết quả : Sin(x) = 0.
- Kiểm thử 2: x = π/2 (≈ 1.5708), kết quả : Sin(x) = 1.
- Kiểm thử 3: x = π (≈ 3.1416), kết quả : Sin(x) = -0.00001 ≈ 0 .
- Kiểm thử 4: x = 3π/2 (≈ 4.7124), kết quả : Sin(x) = -1.
- Kiểm thử 5: x = 2π (≈ 6.2832), kết quả : Sin(x) = 0.00002 ≈ 0.
- Kiểm thử 6: x = 9, kết quả : Sin(x) = 0.412119.
- Kiểm thử 7: x = 0.5, kết quả : Sin(x) = 0.479426.
- Kiểm thử 8: x = -π (≈ -3.1416), kết quả : Sin(x) = 0.00001 ≈ 0.

Bài tập 5: Viết chương trình nhập vào một ngày.

- Tìm ngày kế tiếp và xuất kết quả.
- Tìm ngày trước ngày này và xuất kết quả.
- Tính xem ngày đó là ngày thứ bao nhiêu trong năm và xuất kết quả.

Nội dung flowchart của chương trình như hình 5.



5. Hình 5

Mô tả đầu vào, đầu ra và hướng giải quyết của chương trình:

- Đầu vào: ngày, tháng, năm.
- Đầu ra: Ngày kế tiếp, ngày trước đó và ngày thứ bao nhiêu trong năm.
- Hướng giải quyết: Kiểm tra năm nhuận, tháng 2, các tháng có 30/31 ngày. Dựa vào đó tìm ngày kế tiếp, ngày trước đó và tính tổng các ngày trước đó để tìm thứ tự ngày trong năm.

Nội dung code của chương trình:

```
#include <iostream>

using namespace std;
```

```
//Khai báo cấu trúc Ngay. Khởi tạo giá trị mặc định cho ngày, tháng, năm ( Tránh lỗi phát sinh hàm bool ).
struct Ngay{
	int dd =1, mm=1, yy=1;
};

/*Hàm kiểm tra năm nhuận:
Đầu vào: một số nguyên.
Đầu ra: nếu là năm nhuận, trả về true, không nhuận trả về false*/
bool namNhuan(int nam) {
	if ((nam % 4 == 0 && nam % 100 != 0) || nam % 400 == 0) {
		return true;
	}
	return false;
}

/*Hàm kiểm tra ngày hợp lệ:
Đầu vào: giá trị của biến có kiểu Ngay.
Đầu ra: Nếu ngày tháng nhập vào không thỏa các điều kiện sẽ trả về false, hợp lệ trả về true.*/
bool ngayHopLe(Ngay ngay) {
	if (ngay.mm < 1 || ngay.mm > 12) {
		return false;
	}
	if (ngay.dd < 1 || ngay.dd > 31) {
```

```
                return false;
        }
        if (ngay.mm == 4 || ngay.mm == 6 || ngay.mm == 9 || ngay.mm == 11) {
                if (ngay.dd > 30) {
                        return false;
                }
        }
        if (ngay.mm == 2) {
                if (namNhuan(ngay.yy)) {
                        if (ngay.dd > 29) {
                                return false;
                        }
                }
                else {
                        if (ngay.dd > 28) {
                                return false;
                        }
                }
        }
        return true;
}


/*Hàm nhập ngày:
Đầu vào: giá trị của biến có kiểu Ngay. Nếu không thỏa điều kiện sẽ nhập lại.
Đầu ra: lưu giá trị thông qua biến tham chiếu.*/
```

```
void Nhap(Ngay &ngay) {
	do{
		if (!ngayHopLe(ngay)) {
			cout << "Ngay thang nam khong hop le. Vui long nhap lai." << endl;
		}
		cout << "Nhap ngay: ";
		cin >> ngay.dd;
		cout << "Nhap thang: ";
		cin >> ngay.mm;
		cout << "Nhap nam: ";
		cin >> ngay.yy;
	} while (!ngayHopLe(ngay));
}

/*Hàm tìm ngày kế tiếp:
Đầu vào: giá trị của biến có kiểu Ngay.
Đầu ra: In ra màn hình ngày kế tiếp.
Kiểm tra năm nhuận cho tháng 2,
ngày cuối cùng của tháng,
số ngày của các tháng để xác định ngày kế tiếp.*/
void ngayKeTiep(Ngay ngay) {
	if (ngay.mm == 12 && ngay.dd == 31) {
		cout << "Ngay ke tiep la: 01/01/" << ngay.yy + 1 << endl;
	}
	else if (ngay.mm == 2 && ngay.dd == 28 && !namNhuan(ngay.yy)) {
```

```
            cout << "Ngay ke tiep la: 01/03/" << ngay.yy << endl;
        }
        else if (ngay.mm == 2 && ngay.dd == 29 && namNhuan(ngay.yy)) {
            cout << "Ngay ke tiep la: 01/03/" << ngay.yy << endl;
        }
        else if ((ngay.mm == 4 || ngay.mm == 6 || ngay.mm == 9 || ngay.mm == 11) &&
ngay.dd == 30) {
            cout << "Ngay ke tiep la: 01/" << ngay.mm + 1 << "/" << ngay.yy << endl;
        }
        else if (ngay.dd == 31) {
            cout << "Ngay ke tiep la: 01/" << ngay.mm + 1 << "/" << ngay.yy << endl;
        }
        else {
            cout << "Ngay ke tiep la: " << ngay.dd + 1 << "/" << ngay.mm << "/" <<
ngay.yy << endl;
        }
}


/*Hàm tìm ngày trước đó:
Đầu vào: giá trị của biến có kiểu Ngay.
Đầu ra: In ra màn hình ngày trước đó.
Kiểm tra năm nhuận cho tháng 2,
ngày đầu tiên của tháng,
số ngày của các tháng để xác định ngày trước đó.*/
void ngayTruocDo(Ngay ngay) {
```

```
	if (ngay.mm == 1 && ngay.dd == 1) {
		cout << "Ngay truoc do la: 31/12/" << ngay.yy - 1 << endl;
	}
	else if (ngay.mm == 3 && ngay.dd == 1) {
		if (namNhuan(ngay.yy)) {
			cout << "Ngay truoc do la: 29/02/" << ngay.yy << endl;
		}
		else {
			cout << "Ngay truoc do la: 28/02/" << ngay.yy << endl;
		}
	}
	else if ((ngay.mm == 5 || ngay.mm == 7 || ngay.mm == 10 || ngay.mm == 12) && ngay.dd == 1) {
		cout << "Ngay truoc do la: 30/" << ngay.mm - 1 << "/" << ngay.yy << endl;
	}
	else if (ngay.dd == 1) {
		cout << "Ngay truoc do la: 31/" << ngay.mm - 1 << "/" << ngay.yy << endl;
	}
	else {
		cout << "Ngay truoc do la: " << ngay.dd - 1 << "/" << ngay.mm << "/" << ngay.yy << endl;
	}
}


/*Hàm xác định thứ tự ngày trong năm:
```

```
Đầu vào: giá trị của biến có kiểu Ngay.

Đầu ra: In ra màn hình ngày trước đó.

Kiểm tra năm nhuận cho tháng 2,

số ngày của các tháng,

dùng vòng lặp cộng tổng số ngày từ đầu năm tới ngày này.*/

void thuTuInYear(Ngay ngay) {

	int soNgay = 0;

	for (int i = 1; i < ngay.mm; i++) {

		if (i == 4 || i == 6 || i == 9 || i == 11) {

			soNgay += 30;

		}

		else if (i == 2) {

			if (namNhuan(ngay.yy)) {

				soNgay += 29;

			}

			else {

				soNgay += 28;

			}

		}

		else {

			soNgay += 31;

		}

	}

	soNgay += ngay.dd;

	cout << "Day la ngay thu " << soNgay << " trong nam." << endl;
```

```
}

int main() {

	Ngay date;

	Nhap(date);

	ngayKeTiep(date);

	ngayTruocDo(date);

	thuTuInYear(date);

}
```

5.Bảng 5

Nội dung kiểm thử của chương trình:

- Kiểm thử 1: Ngày: 06/04/2006, kết quả: Ngày kế tiếp: 07/04/2006, ngày trước đó: 05/04/2006, đây là ngày thứ 96 trong năm.
- Kiểm thử 2: Ngày: 01/04/2006, kết quả: Ngày kế tiếp: 02/04/2006, ngày trước đó: 31/03/2006, đây là ngày thứ 91 trong năm.
- Kiểm thử 3: Ngày: 30/04/2006, kết quả: Ngày kế tiếp: 01/05/2006, ngày trước đó: 29/04/2006, đây là ngày thứ 120 trong năm.
- Kiểm thử 4: Ngày: 31/05/2006, kết quả: Ngày kế tiếp: 01/06/2006, ngày trước đó: 30/05/2006, đây là ngày thứ 151 trong năm.
- Kiểm thử 5: Ngày: 01/03/2006, kết quả: Ngày kế tiếp: 02/03/2006, ngày trước đó: 28/02/2006, đây là ngày thứ 60 trong năm.
- Kiểm thử 6: Ngày: 01/03/2008, kết quả: Ngày kế tiếp: 01/05/2008, ngày trước đó: 29/02/2008, đây là ngày thứ 61 trong năm.
- Kiểm thử 7: Ngày: 31/12/2024, kết quả: Ngày kế tiếp: 01/01/2025, ngày trước đó: 30/12/2024, đây là ngày thứ 366 trong năm.
- Kiểm thử 8: Ngày: 01/01/2024, kết quả: Ngày kế tiếp: 02/01/2024, ngày trước đó: 31/12/2023, đây là ngày thứ 1 trong năm.
- Kiểm thử 9: Ngày: 31/02/2024, kết quả: Nhập lại ngày tháng năm.

Bài tập 6: Cho hai mảng A và B chứa các số nguyên. Hãy xây dựng một chương trình để xác định xem có bao nhiêu lần mảng A xuất hiện liên tiếp dưới dạng một dãy con trong mảng B. Cụ thể: Mảng A được coi là xuất hiện trong mảng B nếu tất cả các phần tử của A xuất hiện liên tiếp theo đúng thứ tự trong B. Các vị trí xuất hiện của A trong B có thể chồng lấn lên nhau. Yêu cầu:

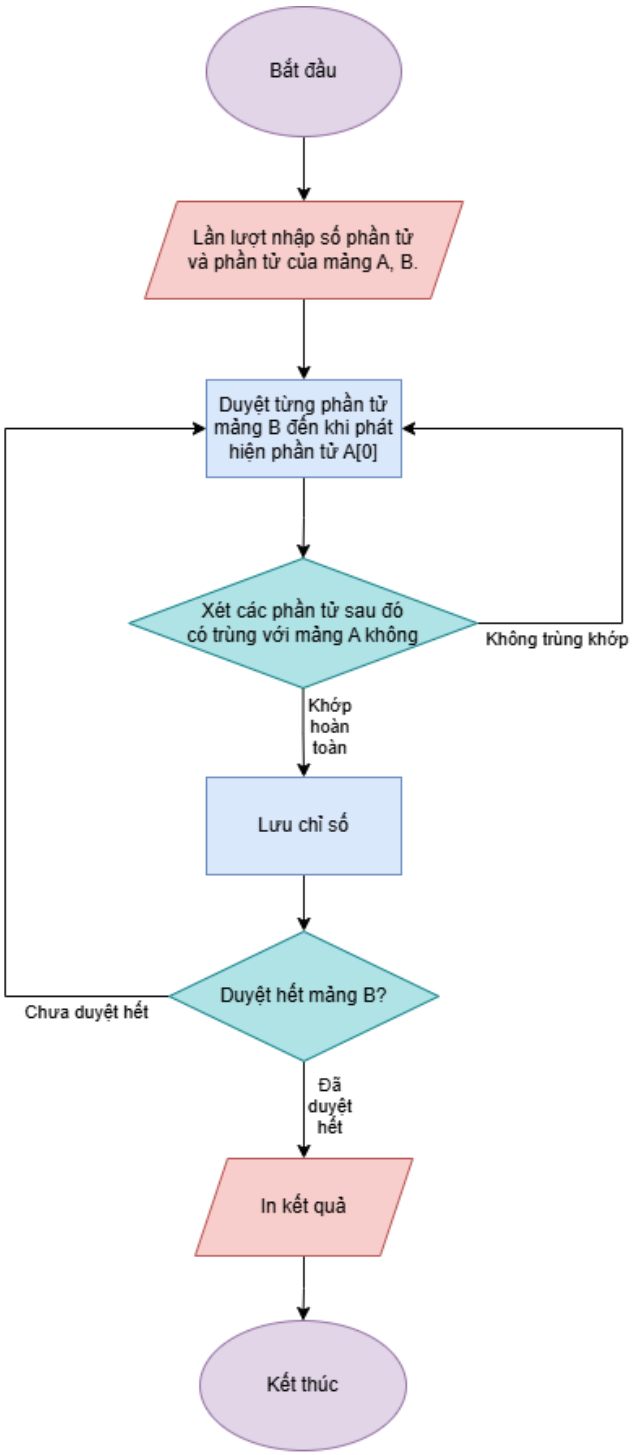
1. Viết một thuật toán hiệu quả để đếm số lần xuất hiện của A trong B.

2. Đảm bảo rằng chương trình có thể xử lý các trường hợp mảng lớn và mảng con xuất hiện nhiều lần liên tiếp trong mảng cha.

3. Trong phần xuất kết quả, liệt kê tất cả các vị trí bắt đầu (chỉ số) của các lần xuất hiện của A trong B.

Ví dụ: Với A = [1, 2], B = [1, 2, 1, 2, 3, 1, 2], mảng A xuất hiện 3 lần tại các chỉ số bắt đầu là 0, 2, và 5.

Nội dung flowchart của chương trình như hình 6.



6. Hình 6

Mô tả đầu vào, đầu ra và hướng giải quyết của chương trình:

- Đầu vào: Dòng đầu tiên chứa số phần tử của mảng A và mảng B. Dòng thứ hai chứa các phần tử của mảng A. Dòng thứ ba chứa các phần tử của mảng B.
- Đầu ra: Số lần mảng A xuất hiện trong mảng B. Danh sách các chỉ số bắt đầu của những lần xuất hiện.

- Hướng giải quyết: Duyệt từng phần tử trong mảng B, ghi nhận chỉ số khi gặp phần tử trùng với phần tử đầu tiên trong mảng A, kiểm tra xem các phần tử sau đó có trùng khớp với các phần tử tiếp theo trong mảng A hay không, nếu không sẽ không lưu chỉ số. Số lần xuất hiện sẽ bằng số chỉ số đã lưu.

Nội dung code của chương trình:

```
#include <iostream>

#define MAX 100

using namespace std;



/*Hàm nhập mảng:

Đầu vào: số phần tử và giá trị của từng phần tử trong mảng a, b.

Đầu ra: lưu phần tử mảng a, b và số phần tử của mảng thông qua biến tham chiếu.*/

void Nhap(int a[], int b[], int& m, int& n) {

	cout << "Nhap lan luot so phan tu cua mang A va B: ";

	cin >> m >> n;

	cout << "Nhap mang A: ";

	for (int i = 0; i < m; i++) {

		cin >> a[i];

	}

	if (m == 0) {

		cout << endl;

	}

	cout << "Nhap mang B: ";

	for (int i = 0; i < n; i++) {

		cin >> b[i];

	}
```

```
}
/*Hàm kiểm tra mảng A có xuất hiện trong mảng B không:
Đầu vào: mảng a, b và số phần tử của mảng a, b.
Đầu ra: số lần mảng a xuất hiện trong mảng b và lưu vị trí bắt đầu của mảng a trong mảng b.*/
void listChiSo(int a[], int b[], int m, int n, int LCS[], int &dem) {
	int chiSoB = 0, chiSoLCS = 0;
	while (chiSoB < n) {
		bool streak = true;
		if (a[0] == b[chiSoB]) {
			streak = true;
			for (int i = 0; i < m; i++) {
				if (a[i] == b[chiSoB + i]) {
					streak = true;
				}
				else {
					streak = false;
					break;
				}
			}
			if (streak) {
				LCS[chiSoLCS] = chiSoB;
				dem++;
				chiSoLCS++;
				streak = true;
```

```
                        }
                }
                chiSoB++;
        }
}

/* Hàm in ra các chỉ số bắt đầu của mảng A trong mảng B:
Đầu vào: mảng LCS(List Chỉ Số) và số lần mảng A xuất hiện trong mảng B.
Đầu ra: in ra các chỉ số bắt đầu của mảng A trong mảng B.*/
void chiSo(int LCS[], int dem) {
        for (int i = 0; i < dem; i++) {
                cout << LCS[i] << " ";
        }
}

int main() {
        int a[MAX], b[MAX],LCS[MAX], m, n, dem=0;
        Nhap(a, b, m, n);
        listChiSo(a, b, m, n, LCS, dem);
        cout << "So lan xuat hien cua mang A trong mang B la: " << dem << endl;
        if (dem != 0) {
                cout << "Chi so bat dau cua nhung lan xuat hien la: ";
                chiSo(LCS, dem);
        }
}
```

6.Bảng 6

Nội dung kiểm thử của chương trình:

- Kiểm thử 1: Nhập A = [1, 2], B = [1, 2, 1, 2, 3, 1, 2], Kết quả: A xuất hiện 3 lần tại 0, 2 và 5.
- Kiểm thử 2: Nhập A = [4, 5], B = [1, 2, 3, 4, 5], Kết quả: A xuất hiện 1 lần tại 3.
- Kiểm thử 3: Nhập A = [9, 9], B = [1, 2, 3, 4, 5], Kết quả: A xuất hiện 0 lần.
- Kiểm thử 4: Nhập A = [1, 2, 3], B = [1, 2, 3], Kết quả: A xuất hiện 1 lần tại 0.
- Kiểm thử 5: Nhập A = [2, 2], B = [2, 2, 2, 2, 2, 2], Kết quả: A xuất hiện 5 lần tại 0, 1, 2, 3 và 4.
- Kiểm thử 6: Nhập A = [3, 4, 5], B = [1, 2, 3, 4, 5, 3, 4, 5], Kết quả: A xuất hiện 2 lần tại 2 và 5.
- Kiểm thử 7: Nhập A = [6, 7], B = [6, 7, 6, 7, 6, 7, 6, 7], Kết quả: A xuất hiện 4 lần tại 0, 2, 4 và 6.
- Kiểm thử 8: Nhập A = [], B = [1, 2, 3, 4, 5], Kết quả: A xuất hiện 0 lần.

Link code: UIT24521056/Lab1-IT002.P214.2.