// 1. Làm lại bài số phức với một phương thức thiết lập duy nhất cho phép quan điểm một số thực như một số phức đặc biệt (phần ảo bằng 0). Định nghĩa các phép toán +, -, \*, /, = =, !=, >, >=, <, <= trên số phức. Định nghĩa phép toán << và >> để xuất và nhập dữ liệu vào số phức.

#include <bits/stdc++.h>

using *namespace* std;

*class* *ComplexNum* {

*private:*

*double* Thuc, Ao;

*public:*

        ComplexNum(){

            this->Thuc = 0;

            this->Ao = 0;

        }

        ComplexNum(*double* *real*, *double* *imag*){

            Thuc = *real*;

            Ao = *imag*;

        }

        friend *istream*& operator >> (*istream*& *in*, *ComplexNum*& *x*){

*in* >> *x*.Thuc >> *x*.Ao;

            return *in*;

        }

        friend *ostream*& operator << (*ostream*& *out*, *ComplexNum* *x*){

*out* << *x*.Thuc;

            if (*x*.Ao >= 0) *out* << " + " << *x*.Ao << "i";

            else *out* << " - " << -*x*.Ao << "i";

            return *out*;

        }

*ComplexNum* operator + (*ComplexNum* *x*){

            return *ComplexNum*(*x*.Thuc + Thuc, *x*.Ao + Ao);

        }

*ComplexNum* operator - (*ComplexNum* *x*){

            return *ComplexNum*(Thuc - *x*.Thuc, Ao - *x*.Ao);

        }

*ComplexNum* operator \* (*ComplexNum* *x*){

            return *ComplexNum*(Thuc\**x*.Thuc - Ao\**x*.Ao, Thuc\**x*.Ao + Ao\**x*.Thuc);

        }

*ComplexNum* operator / (*ComplexNum* *x*){

            return *ComplexNum*((Thuc\**x*.Thuc + Ao\**x*.Ao)/(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao), (Ao\**x*.Thuc - *x*.Ao\*Thuc)/(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao));

        }

*bool* operator == (*ComplexNum* *x*){

            if(*x*.Thuc == Thuc && *x*.Ao == Ao) return true;

            return false;

        }

*bool* operator != (*ComplexNum* *x*){

            if(*x*.Thuc != Thuc || *x*.Ao != Ao) return true;

            return false;

        }

*bool* operator > (*ComplexNum* *x*){

            if(sqrt(Thuc\*Thuc + Ao\*Ao) > sqrt(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao)) return true;

            return false;

        }

*bool* operator >= (*ComplexNum* *x*){

            if(sqrt(Thuc\*Thuc + Ao\*Ao) > sqrt(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao) || *x* == \*this) return true;

            return false;

        }

*bool* operator < (*ComplexNum* *x*){

            if(sqrt(Thuc\*Thuc + Ao\*Ao) < sqrt(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao)) return true;

            return false;

        }

*bool* operator <= (*ComplexNum* *x*){

            if(sqrt(Thuc\*Thuc + Ao\*Ao) < sqrt(*x*.Thuc\**x*.Thuc + *x*.Ao\**x*.Ao) || *x* == \*this) return true;

            return false;

        }

*double* getThuc(){

            return this->Thuc;

        }

*double* getAo(){

            return this->Ao;

        }

};

*int* main(){

*ComplexNum* x, y;

    cout << "Nhap so phuc thu nhat: ";

    cin >> x;

    cout << "Nhap so phuc thu hai: ";

    cin >> y;

    cout << "Tong hai so phuc la: " << x + y << endl;

    cout << "Hieu hai so phuc la: " << x - y << endl;

    cout << "Tich hai so phuc la: " << x \* y << endl;

    if(y.getThuc() == 0 && y.getAo() == 0)

        cout << "Loi chia cho 0!" << endl;

    else

        cout << "Thuong hai so phuc la: " << x / y << endl;

    if(x == y)

        cout << "Hai so phuc bang nhau!" << endl;

    else if(x < y)

        cout << "So phuc thu nhat be hon so phuc thu hai!" << endl;

    else

        cout << "So phuc thu nhat lon hon so phuc thu hai!" << endl;

    return 0;

}

// 2. Làm lại bài phân số với các phương thức thiết lập cho phép sử dụng một số nguyên như một phân số đặc biệt (mẫu số bằng 1). Định nghĩa các phép toán +, -, \*, /, = =, != , >, >=, <, <= trên phân số. Định nghĩa phép toán << và >> để xuất và nhập dữ liệu vào phân số.

#include <bits/stdc++.h>

using *namespace* std;

*int* gcd(*int* *a*, *int* *b*){

    if(*b* == 0) return *a*;

    return gcd(*b*, *a*%*b*);

}

*class* *PhanSo* {

*private:*

*int* Tu, Mau;

*public:*

        PhanSo(){

            this->Tu = 0;

            this->Mau = 1;

        }

        PhanSo(*int* *Tu*, *int* *Mau*){

            this->Tu = *Tu*;

            this->Mau = *Mau*;

        }

        friend *istream*& operator >> (*istream*& *in*, *PhanSo*& *x*){

*in* >> *x*.Tu >> *x*.Mau;

            while(*x*.Mau == 0){

                cout << "Loi mau so la 0, moi nhap so khac: ";

*in* >> *x*.Tu >> *x*.Mau;

            }

            return *in*;

        }

        friend *ostream*& operator << (*ostream*& *out*, *PhanSo* *x*){

            if(*x*.Mau == 1)

*out* << *x*.Tu;

            else if(*x*.Tu == 0)

*out* << 0;

            else if(*x*.Mau < 0)

*out* << -*x*.Tu << "/" << -*x*.Mau;

            else

*out* << *x*.Tu << "/" << *x*.Mau;

            return *out*;

        }

*PhanSo* operator + (*PhanSo* *x*){

            return *PhanSo*(Tu\**x*.Mau + *x*.Tu\*Mau, Mau\**x*.Mau);

        }

*PhanSo* operator - (*PhanSo* *x*){

            return *PhanSo*(Tu\**x*.Mau - *x*.Tu\*Mau, Mau\**x*.Mau);

        }

*PhanSo* operator \* (*PhanSo* *x*){

            return *PhanSo*(Tu\**x*.Tu, Mau\**x*.Mau);

        }

*PhanSo* operator / (*PhanSo* *x*){

            return *PhanSo*(Tu\**x*.Mau, Mau\**x*.Tu);

        }

*bool* operator == (*PhanSo* *x*){

            return Tu \* *x*.Mau == *x*.Tu \* Mau;

        }

*bool* operator != (*PhanSo* *x*){

            return !(\*this == *x*);

        }

*bool* operator > (*PhanSo* *x*){

            return Tu \* *x*.Mau > *x*.Tu \* Mau;

        }

*bool* operator >= (*PhanSo* *x*){

            return Tu \* *x*.Mau >= *x*.Tu \* Mau;

        }

*bool* operator < (*PhanSo* *x*){

            return Tu \* *x*.Mau < *x*.Tu \* Mau;

        }

*bool* operator <= (*PhanSo* *x*){

            return Tu \* *x*.Mau <= *x*.Tu \* Mau;

        }

*int* getTu() const {

            return this->Tu;

        }

*int* getMau() const {

            return this->Mau;

        }

};

*PhanSo* RutGon(*PhanSo* *x*){

*int* tu = *x*.getTu()/gcd(*x*.getTu(), *x*.getMau());

*int* mau = *x*.getMau()/gcd(*x*.getTu(), *x*.getMau());

    return *PhanSo*(tu, mau);

}

*int* main(){

*PhanSo* x, y;

    cout << "Nhap phan so thu nhat: ";

    cin >> x;

    cout << "Nhap phan so thu hai: ";

    cin >> y;

    cout << "Tong hai phan so la: " << RutGon(x + y) << endl;

    cout << "Hieu hai phan so la: " << RutGon(x - y) << endl;

    cout << "Tich hai phan so la: " << RutGon(x \* y) << endl;

    if(y.getTu() == 0)

        cout << "Loi chia cho so 0" << endl;

    else

        cout << "Thuong hai phan so la: " << RutGon(x / y)  << endl;

    if(x == y)

        cout << x << " = " << y << endl;

    if(x < y)

        cout << x << " < " << y << endl;

    if(x <= y)

        cout << x << " <= " << y << endl;

    if(x > y)

        cout << x << " > " << y << endl;

    if(x >= y)

        cout << x << " >= " << y << endl;

    return 0;

}

// 3. Định nghĩa lớp dữ liệu CTimeSpan để biểu diễn khái niệm khoảng thời gian, các hàm thành phần và các phép toán cần thiết (+, -, ==, !=, >, >=, <, <=).

#include <bits/stdc++.h>

using *namespace* std;

*class* *CTimeSpan* {

*private:*

*int* Hour, Minute, Second;

*public:*

        CTimeSpan(){

            Hour = Minute = Second = 0;

        }

        CTimeSpan(*int* *h*, *int* *m*, *int* *s*){

            this->Hour = *h*;

            this->Minute = *m*;

            this->Second = *s*;

        }

        friend *istream*& operator >> (*istream*& *in*, *CTimeSpan*& *x*){

*in* >> *x*.Hour >> *x*.Minute >> *x*.Second;

            return *in*;

        }

        friend *ostream*& operator << (*ostream*& *out*, *CTimeSpan* *x*){

*out* << *x*.Hour << "h " << *x*.Minute << "m " << *x*.Second << "s ";

            return *out*;

        }

*CTimeSpan* operator + (*CTimeSpan* *x*){

*int* sumHours = Hour + *x*.Hour;

*int* sumMinutes = Minute + *x*.Minute;

*int* sumSeconds = Second + *x*.Second;

            if(sumSeconds >= 60){

                sumSeconds = sumSeconds - 60;

                sumMinutes++;

            }

            if(sumMinutes >= 60){

                sumMinutes = sumMinutes - 60;

                sumHours++;

            }

            if(sumHours >= 24) sumHours = sumHours - 24;

            return *CTimeSpan*(sumHours, sumMinutes, sumSeconds);

        }

*CTimeSpan* operator - (*CTimeSpan* *x*){

*int* newsec = *x*.Second, newmin = *x*.Minute, newhours = *x*.Hour;

            if(Second > *x*.Second){

                newmin--;

                newsec += 60;

*x*.Minute = newmin;

*x*.Second = newsec;

            }

            if(*x*.Minute < Minute){

                newhours--;

                newmin += 60;

*x*.Hour = newhours;

*x*.Minute = newmin;

            }

*int* minusHours = *x*.Hour - Hour;

*int* minusMinutes = *x*.Minute - Minute;

*int* minusSeconds = *x*.Second - Second;

            if(minusHours < 0) minusHours = 0;

            if(minusMinutes < 0) minusMinutes = 0;

            if(minusSeconds < 0) minusSeconds = 0;

            return *CTimeSpan*(minusHours, minusMinutes, minusSeconds);

        }

*bool* operator == (*CTimeSpan* *x*){

            if(*x*.Hour == Hour && *x*.Minute == Minute && *x*.Second == Second) return true;

            return false;

        }

*bool* operator != (*CTimeSpan* *x*){

            if(*x*.Hour != Hour || *x*.Minute != Minute || *x*.Second != Second) return true;

            return false;

        }

*int* ChangeTime(*CTimeSpan* *x*){

*int* time = *x*.Hour\*3600 + *x*.Minute\*60 + *x*.Second;

            return time;

        }

*bool* operator > (*CTimeSpan* *x*){

            if(ChangeTime(*x*) < ChangeTime(\*this)) return true;

            return false;

        }

*bool* operator >= (*CTimeSpan* *x*){

            if(ChangeTime(*x*) < ChangeTime(\*this) || *x* == \*this) return true;

            return false;

        }

*bool* operator < (*CTimeSpan* *x*){

            if(ChangeTime(*x*) > ChangeTime(\*this)) return true;

            return false;

        }

*bool* operator <= (*CTimeSpan* *x*){

            if(ChangeTime(*x*) > ChangeTime(\*this) || *x* == \*this) return true;

            return false;

        }

};

*int* main(){

*CTimeSpan* x, y;

    cout << "Nhap thoi gian thu nhat: ";

    cin >> x;

    cout << "Nhap thoi gian thu hai: ";

    cin >> y;

    cout << "Tong hai thoi gian la: " << x + y << endl;

    cout << "Hieu hai thoi gian la: " << x - y << endl;

    if(x < y) cout << x << " < " << y << endl;

    else if(x <= y) cout << x << " <= " << y << endl;

    if(x > y) cout << x << " > " << y << endl;

    else if(x >= y) cout << x << " >= " << y << endl;

}

// 4. Định nghĩa lớp CTime biểu diễn khái niệm thời điểm có các thành phần giờ phút giây. Định nghĩa các phép toán +, - (cộng, trừ thêm một số nguyên giây), - (phép trừ hai CTime để được một CTimSpan), ++, -- (thêm bớt một giây). Phép toán <<, >> để xuất, nhập dữ liệu loại CTime.

#include <bits/stdc++.h>

using *namespace* std;

*class* *CTime* {

*private:*

*int* Hour, Minute, Second;

*public:*

    CTime(){

        Hour = Minute = Second = 0;

    }

    CTime(*int* *h*, *int* *m*, *int* *s*){

        this->Hour = *h*;

        this->Minute = *m*;

        this->Second = *s*;

    }

    friend *istream*& operator >> (*istream*& *in*, *CTime*& *x*){

*in* >> *x*.Hour >> *x*.Minute >> *x*.Second;

        return *in*;

    }

    friend *ostream*& operator << (*ostream*& *out*, *CTime* *x*){

*out* << *x*.Hour << "h " << *x*.Minute << "m " << *x*.Second << "s ";

        return *out*;

    }

*int* Changtime(){

        return Hour\*3600 + Minute\*60 + Second;

    }

*CTime* Settime(){

*int* time = Changtime();

        while (time < 0)

            time += 24 \* 3600;

        while (time >= 24 \* 3600)

            time -= 24 \* 3600;

        Hour = time / 3600;

        time = time - Hour \* 3600;

        Minute = time / 60;

        Second = time - Minute \* 60;

        return \*this;

    }

*CTime* operator + (*int* *news*){

        return *CTime*(Hour, Minute, Second + *news*).Settime();

    }

*CTime* operator - (*int* *news*){

        return *CTime*(Hour, Minute, Second - *news*).Settime();

    }

*CTime* operator - (*CTime* *x*){

*CTime* hieu;

        hieu.Hour = Hour - *x*.Hour;

        hieu.Minute = Minute - *x*.Minute;

        hieu.Second = Second - *x*.Second;

        hieu.Settime();

        return hieu;

    }

    // ++x

*CTime*& operator ++ (){

        Second++;

        if(Second > 60 || Second < 0) Settime();

        return \*this;

    }

    // x++

*CTime* operator ++ (*int*){

*CTime* t = \*this;

        Second++;

        if (Second > 60 || Second < 0) Settime();

        return t;

    }

    // --x

*CTime*& operator -- (){

        Second--;

        if(Second > 60 || Second < 0) Settime();

        return \*this;

    }

    // x--

*CTime* operator -- (*int*){

*CTime* t = \*this;

        Second--;

        if (Second > 60 || Second < 0) Settime();

        return t;

    }

};

*int* main(){

*CTime* x, y;

*int* n;

    cout << "Nhap so nguyen giay: ";

    cin >> n;

    cout << "Nhap thoi gian thu nhat: ";

    cin >> x;

    cout << "Nhap thoi gian thu hai: ";

    cin >> y;

    cout << "Thoi gian moi sau khi cong them so nguyen giay la: " << x + n << " " << y + n << endl;

    cout << "Thoi gian moi sau khi tru them so nguyen giay la: " << x - n << " " << y - n << endl;

    cout << "Hieu hai thoi gian la: " << x - y << endl;

    cout << "Time++ " << x++ << " " << y++ << endl;

    cout << "++Time " << ++x << " " << ++y << endl;

    cout << "Time-- " << x-- << " " << y-- << endl;

    cout << "--Time " << --x << " " << --y << endl;

    return 0;

}

// 5. Định nghĩa lớp CDate biểu diễn khái niệm ngày, tháng, năm với các phép toán +, - (cộng, trừ thêm một số ngày), ++, -- (thêm bớt một ngày), - (khoảng cách giữa hai CDate tính bằng ngày). Phép toán <<, >> để xuất, nhập dữ liệu loại CDate.

#include <bits/stdc++.h>

using *namespace* std;

*class* *CDate* {

*private:*

*int* Year, Month, Day;

*public:*

    CDate(){

        Year = Month = Day = 0;

    }

    CDate(*int* *y*, *int* *m*, *int* *d*){

        this->Year = *y*;

        this->Month = *m*;

        this->Day = *d*;

    }

    friend *istream*& operator >> (*istream*& *in*, *CDate*& *x*){

*in* >> *x*.Day >> *x*.Month >> *x*.Year;

        return *in*;

    }

    friend *ostream*& operator << (*ostream*& *out*, *CDate* *x*){

*out* << *x*.Day << "/" << *x*.Month << "/" << *x*.Year;

        return *out*;

    }

*bool* isleapYear(){

        if(Year % 4 == 0){

            if(Year % 100 != 0) return true;

            else{

                if(Year % 400 == 0) return true;

                else return false;

            }

        }

        return false;

    }

*CDate* SetDate() {

*int* daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        if (isleapYear()) {

            daysInMonth[1] = 29;

        }

        while (Day > daysInMonth[Month - 1]) {

            Day -= daysInMonth[Month - 1];

            Month++;

            if (Month > 12) {

                Month = 1;

                Year++;

            }

        }

        while (Day <= 0) {

            Month--;

            if (Month <= 0) {

                Month = 12;

                Year--;

            }

            Day += daysInMonth[Month - 1];

        }

        return \*this;

    }

*int* ToDate(*CDate* *x*){

*int* ChangeTime;

        if(isleapYear()){

            if(*x*.Month == 1 || *x*.Month == 3 || *x*.Month == 5 || *x*.Month == 7 || *x*.Month == 8 || *x*.Month == 10 || *x*.Month == 12){

                ChangeTime = *x*.Year\*366 + *x*.Month\*31 + *x*.Day;

            }

            else if(*x*.Month == 4 || *x*.Month == 6 || *x*.Month == 9 || *x*.Month == 11){

                ChangeTime = *x*.Year\*366 + *x*.Month\*30 + *x*.Day;

            }

            else if(*x*.Month == 2){

                ChangeTime = *x*.Year\*366 + *x*.Month\*29 + *x*.Day;

            }

        }

        else{

            if(*x*.Month == 1 || *x*.Month == 3 || *x*.Month == 5 || *x*.Month == 7 || *x*.Month == 8 || *x*.Month == 10 || *x*.Month == 12){

                ChangeTime = *x*.Year\*365 + *x*.Month\*31 + *x*.Day;

            }

            else if(*x*.Month == 4 || *x*.Month == 6 || *x*.Month == 9 || *x*.Month == 11){

                ChangeTime = *x*.Year\*365 + *x*.Month\*30 + *x*.Day;

            }

            else if(*x*.Month == 2){

                ChangeTime = *x*.Year\*365 + *x*.Month\*28 + *x*.Day;

            }

        }

        return ChangeTime;

    }

*CDate* operator + (*int* *n*){

*CDate* temp = \*this;

        temp.Day += *n*;

        temp.SetDate();

        return temp;

    }

*CDate* operator - (*int* *n*){

*CDate* temp = \*this;

        temp.Day -= *n*;

        temp.SetDate();

        return temp;

    }

*int* operator - (*CDate* *x*){

*int* A = ToDate(*x*);

*int* B = ToDate(\*this);

        return abs(A - B);

    }

    // ++x, --x;

*CDate*& operator ++ (){

        Day++;

        SetDate();

        return \*this;

    }

*CDate*& operator -- (){

        Day--;

        SetDate();

        return \*this;

    }

    // x++, x--;

*CDate* operator ++ (*int*){

*CDate* temp = \*this;

        Day++;

        SetDate();

        return temp;

    }

*CDate* operator -- (*int*){

*CDate* temp = \*this;

        Day--;

        SetDate();

        return temp;

    }

};

*int* main(){

*CDate* x, y, origin1, origin2;

*int* n;

    cout << "Nhap CDate thu nhat: ";

    cin >> x;

    cout << "Nhap CDate thu hai: ";

    cin >> y;

    origin1 = x;

    origin2 = y;

    cout << "Nhap n ngay ban muon them vao: ";

    cin >> n;

    cout << "CDate sau khi cong them n ngay la: " << x + n << " " << y + n << endl;

    cout << "CDate sau khi tru them n ngay la: " << x - n << " " << y - n << endl;

    cout << "Khoang cach hai CDate la: " << x - y << endl;

    cout << "++Time la: " << ++x << " " << ++y << endl;

    x = origin1;

    y = origin2;

    cout << "Time++ la: " << x++ << " " << y++ << endl;

    x = origin1;

    y = origin2;

    cout << "--Time la: " << --x << " " << --y << endl;

    x = origin1;

    y = origin2;

    cout << "Time-- la: " << x-- << " " << y-- << endl;

    return 0;

}

// 6. Hãy định nghĩa lớp CString biểu diễn khái niệm chuỗi ký tự với các phương thức thiết lập, huỷ bỏ, các hàm thành phần và các phép toán cần thiết (+, gán, so sánh hai chuỗi).

#include <iostream>;

#include <string>

using namespace std;

class CString {

    string s;

public:

    CString(){

        s = "";

    }

    friend istream& operator>>(istream& is, CString& a);

    friend ostream& operator<<(ostream& os, CString& a);

    void setString(string value);

    CString operator+(CString value);

    bool operator!=(CString value);

    ~CString() {}

};

int main() {

    CString s1, s2;

    cin >> s1 >> s2;

    CString total = s1 + s2;

    cout << s1 << " + " << s2 << " = " << total << endl;

    if (s1 != s2)

        cout << s1 << " != " << s2 << endl;

    else cout << s1 << " = " << s2 << endl;

}

istream& operator>>(istream& is, CString& a) {

    cout << "Xau nhap vao la ";

    is >> a.s;

    return is;

}

ostream& operator<<(ostream& os, CString& a) {

    os << a.s;

    return os;

}

void CString::setString(string value) {

    this->s = value;

}

CString CString::operator+(CString value) {

    CString result;

    result.s = s + value.s;

    return result;

}

bool CString::operator!=(CString value) {

    if (s.compare(value.s) != 0)

        return true;

    return false;

}

// 7. Định nghĩa lớp biểu diễn khái niệm đa thức có bậc bất kỳ với các hàm thành phần và phép toán cần thiết.

#include <bits/stdc++.h>

using *namespace* std;

*class* *DaThuc* {

*private:*

*int* bac;

*int* \*arr;

*public:*

        DaThuc(){

            bac = 0;

            arr = new *int*[bac + 1];

            for(*int* i = 0; i<=bac; i++){

                arr[i] = 0;

            }

        }

        DaThuc(const *DaThuc*& *x*){

            bac = *x*.bac;

            arr = new *int*[bac + 1];

            for(*int* i = 0; i<=bac; i++){

                arr[i] = *x*.arr[i];

            }

        }

        friend *istream*& operator >> (*istream*& *in*, *DaThuc*& *a*){

            cout << "\n Hay nhap bac cua da thuc: ";

*in* >> *a*.bac;

*a*.arr = new *int*[*a*.bac + 1];

            for (*int* i = 0; i <= *a*.bac; i++){

                cout << " He so cua x^" << i << " : ";

*in* >> *a*.arr[i];

            }

            return *in*;

        }

        friend *ostream*& operator << (*ostream*& *out*, *DaThuc* *a*){

            for (*int* i = *a*.bac; i > 0; i--){

*out* << *a*.arr[i] << "x^" << i << " + ";

            }

*out* << *a*.arr[0] << endl;

            return *out*;

        }

*DaThuc* operator + (*DaThuc* *b*){

*DaThuc* tong;

            if (bac > *b*.bac){

                tong.bac = bac;

            }

            else{

                tong.bac = *b*.bac;

            }

            delete[]tong.arr;

            tong.arr = new *int*[tong.bac + 1];

            for (*int* i = 0; i <= tong.bac; i++){

                if (i <= bac && i <= *b*.bac) {

                    tong.arr[i] = arr[i] + *b*.arr[i];

                }

                else{

                    if (i > bac) tong.arr[i] = *b*.arr[i];

                    else

                        if (i > *b*.bac) tong.arr[i] = arr[i];

                }

            }

            return tong;

        }

*DaThuc* operator - (*DaThuc* *b*){

            //a-b

*DaThuc* hieu;

            if (bac > *b*.bac){

                hieu.bac = bac;

            }

            else{

                hieu.bac = *b*.bac;

            }

            hieu.arr = new *int*[hieu.bac + 1];

            for (*int* i = 0; i <= hieu.bac; i++){

                if (i <= bac && i <= *b*.bac){

                    hieu.arr[i] = arr[i] - *b*.arr[i];

                }

                else{

                    if (i > bac) hieu.arr[i] = -*b*.arr[i];

                    else

                        if (i > *b*.bac) hieu.arr[i] = arr[i];

                }

            }

            return hieu;

        }

*DaThuc* operator \* (*DaThuc* *b*){

*DaThuc* nhan;

            nhan.bac = bac + *b*.bac;

            nhan.arr = new *int*[nhan.bac + 1];

            for (*int* k = 0; k <= nhan.bac; k++){

                nhan.arr[k] = 0;

            }

            for (*int* i = 0; i <= bac; i++)

                for (*int* j = 0; j <= *b*.bac; j++){

                    nhan.arr[i + j] += arr[i] \* *b*.arr[j];

                }

            return nhan;

        }

};

*int* main(){

*DaThuc* a, b, cong, tru, nhan;

    cout << "\n Nhap da thuc thu nhat F1(x).";

    cin >> a;

    cout << " F1(x) = " << a;

    cout << "\n Nhap da thuc thu hai F2(x).";

    cin >> b;

    cout << " F2(x) = " << b;

    cout << "\n F1(x) + F2(x) = ";

    cong = a + b;

    cout << cong;

    cout << "\n F1(x) - F2(x) = ";

    tru = a - b;

    cout << tru;

    cout << "\n F1(x) \* F2(x) = ";

    nhan = a \* b;

    cout << nhan;

}

// 8. Định nghĩa lớp CVector biểu diễn khái niệm vector trong không gian có số chiều bất kỳ với các hàm thành phần và các phép toán cần thiết.

Định nghĩa lớp CMatrix biểu diễn khái niệm ma trận có kích thước bất kỳ với các hàm thành phần và các phép toán cần thiết.

Viết hàm tính tích của một ma trận và một vector. Tích của hai ma trận.

#include <bits/stdc++.h>

using *namespace* std;

*class* *CVector* {

*private:*

*int* n;

*int*\* arr;

*public:*

    CVector(*int* *n* = 0) {

        this->n = *n*;

        arr = new *int*[*n*];

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

            arr[i] = 0;

        }

    }

*int* getSize() {

        return n;

    }

*int*\* getArr() {

        return arr;

    }

    ~CVector() {

        delete[] arr;

    }

    friend *istream*& operator >> (*istream*& *in*, *CVector*& *x*) {

        cout << "Nhap so chieu cua vector: ";

*in* >> *x*.n;

        delete[] *x*.arr;

*x*.arr = new *int*[*x*.n];

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

            cout << "x" << i + 1 << " = ";

*in* >> *x*.arr[i];

        }

        return *in*;

    }

    friend *ostream*& operator << (*ostream*& *out*, *CVector* *x*) {

*out* << "(";

        for(*int* i = 0; i < *x*.n - 1; i++) {

*out* << *x*.arr[i] << ", ";

        }

*out* << *x*.arr[*x*.n - 1] << ")";

        return *out*;

    }

};

*class* *CMatrix* {

*private:*

*int* n, m;

*int*\*\* arr;

*public:*

    CMatrix(*int* *n* = 0, *int* *m* = 0) {

        this->n = *n*;

        this->m = *m*;

        arr = new *int*\*[*n*];

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

            arr[i] = new *int*[*m*];

        }

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

            for(*int* j = 0; j < *m*; j++) {

                arr[i][j] = 0;

            }

        }

    }

    CMatrix(const *CMatrix*& *other*) {

        n = *other*.n;

        m = *other*.m;

        arr = new *int*\*[n];

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

            arr[i] = new *int*[m];

            for (*int* j = 0; j < m; j++) {

                arr[i][j] = *other*.arr[i][j];

            }

        }

    }

*CMatrix*& operator = (const *CMatrix*& *other*) {

        if (this != &*other*) {

            for (*int* i = 0; i < n; i++) delete[] arr[i];

            delete[] arr;

            n = *other*.n;

            m = *other*.m;

            arr = new *int*\*[n];

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

                arr[i] = new *int*[m];

                for (*int* j = 0; j < m; j++) {

                    arr[i][j] = *other*.arr[i][j];

                }

            }

        }

        return \*this;

    }

    ~CMatrix() {

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

            delete[] arr[i];

        }

        delete[] arr;

    }

    friend *istream*& operator >> (*istream*& *in*, *CMatrix*& *x*) {

        cout << "Nhap so hang va cot cua ma tran: ";

*in* >> *x*.n >> *x*.m;

        if (*x*.n <= 0 || *x*.m <= 0) {

            cout << "Loi: So hang va so cot phai lon hon 0!" << endl;

*x*.n = *x*.m = 0;

            return *in*;

        }

        if (*x*.arr) {

            for (*int* i = 0; i < *x*.n; i++) delete[] *x*.arr[i];

            delete[] *x*.arr;

        }

*x*.arr = new *int*\*[*x*.n];

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

*x*.arr[i] = new *int*[*x*.m];

        }

        cout << "Nhap cac phan tu cua ma tran:\n";

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

            for (*int* j = 0; j < *x*.m; j++) {

*in* >> *x*.arr[i][j];

            }

        }

        return *in*;

    }

    friend *ostream*& operator << (*ostream*& *out*, *CMatrix* *x*){

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

            for(*int* j = 0; j<*x*.m; j++){

*out* << *x*.arr[i][j] << " ";

            }

*out* << endl;

        }

        return *out*;

    }

*CMatrix* operator \* (*CMatrix*& *x*) {

        if (m != *x*.n) {

            cout << "Loi: So cot cua A phai bang so hang cua B!" << endl;

            return *CMatrix*();

        }

*CMatrix* tich(n, *x*.m);

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

            for(*int* j = 0; j < *x*.m; j++) {

                tich.arr[i][j] = 0;

                for (*int* k = 0; k < m; k++)

                    tich.arr[i][j] += arr[i][k] \* *x*.arr[k][j];

            }

        }

        return tich;

    }

};

*int* main(){

*CVector* vt1;

    cout << " Nhap vao mot vecto X\n";

    cin >> vt1;

    cout << " X = " << vt1;

*CMatrix* mt1, mt2, mt3, tich;

    cout << "\n Nhap vao ma tran thu nhat A\n";

    cin >> mt1;

    cout << mt1;

    cout << "\n Nhap vao ma tran thu hai B\n";

    cin >> mt2;

    cout << mt2;

    cout << "A \* B = \n";

    tich = mt1 \* mt2;

    cout << tich << "\n";

    return 0;

}

// 9. Hãy định nghĩa lớp INTEGER có thể hoạt động như để mỗi INTEGER giống hệt như một 'int' của ngôn ngữ C/C++.

#include <iostream>

using namespace std;

class INTERGER {

private:

int x;

public:

INTERGER(int newx = 0);

~INTERGER();

INTERGER& operator++();

INTERGER operator++(int);

INTERGER& operator--();

INTERGER operator--(int);

friend istream& operator>>(istream& is, INTERGER& a);

friend ostream& operator<<(ostream& os, INTERGER a);

friend INTERGER operator+(INTERGER a, INTERGER b);

friend INTERGER operator-(INTERGER a, INTERGER b);

friend INTERGER operator\*(INTERGER a, INTERGER b);

friend INTERGER operator/(INTERGER a, INTERGER b);

friend INTERGER operator%(INTERGER a, INTERGER b);

};

int main(){

INTERGER a, b;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "a + 1 = " << a++ << endl;

cout << "b - 1 = " << --b << endl;

cout << a << " x " << b << " + 9 - " << a << " = " << a \* b + 9 - a << "\n";

cout << a << " / " << b << " = " << a / b << "\n";

cout << a << " % " << b << " = " << a % b << endl;

return 0;

}

INTERGER::INTERGER(int newx) :x(newx)

{

}

INTERGER::~INTERGER()

{

}

INTERGER& INTERGER::operator++()

{

x++;

return \*this;

}

INTERGER INTERGER::operator++(int)

{

INTERGER t = \*this;

x++;

return t;

}

INTERGER& INTERGER::operator--()

{

--x;

return \*this;

}

INTERGER INTERGER::operator--(int)

{

INTERGER t(x);

x--;

return t;

}

istream& operator>>(istream& is, INTERGER& a)

{

is >> a.x;

return is;

}

ostream& operator<<(ostream& os, INTERGER a)

{

os << a.x;

return os;

}

INTERGER operator+(INTERGER a, INTERGER b)

{

return INTERGER(a.x + b.x);

}

INTERGER operator-(INTERGER a, INTERGER b)

{

return INTERGER(a.x - b.x);

}

INTERGER operator\*(INTERGER a, INTERGER b)

{

return INTERGER(a.x \* b.x);

}

INTERGER operator/(INTERGER a, INTERGER b)

{

return INTERGER(a.x / b.x);

}

INTERGER operator%(INTERGER a, INTERGER b)

{

return INTERGER(a.x % b.x);

}

// 10. Hãy định nghĩa lớp MYINT có hoạt động như kiểu dữ liệu 'int' nhưng phép cộng hai MYINT hoạt động như phép trừ hai int và ngược lại.

#include <iostream>

using namespace std;

class MYINT {

private:

int x;

public:

MYINT(int newx = 0);

~MYINT();

MYINT& operator++();

MYINT operator++(int);

MYINT& operator--();

MYINT operator--(int);

friend istream& operator>>(istream& is, MYINT& a);

friend ostream& operator<<(ostream& os, MYINT a);

friend MYINT operator+(MYINT a, MYINT b);

friend MYINT operator-(MYINT a, MYINT b);

friend MYINT operator\*(MYINT a, MYINT b);

friend MYINT operator/(MYINT a, MYINT b);

friend MYINT operator%(MYINT a, MYINT b);

};

int main(){

MYINT a, b;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "a + 1 = " << a++ << endl;

cout << "b - 1 = " << --b << endl;

cout << a << " - " << b << " = " << a - b << "\n";

cout << a << " + " << b << " = " << a + b << "\n";

cout << a << " x " << b << " = " << a \* b << "\n";

cout << a << " / " << b << " = " << a / b << "\n";

cout << a << " % " << b << " = " << a % b << endl;

return 0;

}

MYINT::MYINT(int newx) :x(newx)

{

}

MYINT::~MYINT()

{

}

MYINT& MYINT::operator++()

{

x--;

return \*this;

}

MYINT MYINT::operator++(int)

{

MYINT t = \*this;

x--;

return t;

}

MYINT& MYINT::operator--()

{

++x;

return \*this;

}

MYINT MYINT::operator--(int)

{

MYINT t(x);

x++;

return t;

}

istream& operator>>(istream& is, MYINT& a)

{

is >> a.x;

return is;

}

ostream& operator<<(ostream& os, MYINT a)

{

os << a.x;

return os;

}

MYINT operator+(MYINT a, MYINT b)

{

return MYINT(a.x - b.x);

}

MYINT operator-(MYINT a, MYINT b)

{

return MYINT(a.x + b.x);

}

MYINT operator\*(MYINT a, MYINT b)

{

return MYINT(a.x \* b.x);

}

MYINT operator/(MYINT a, MYINT b)

{

return MYINT(a.x / b.x);

}

MYINT operator%(MYINT a, MYINT b)

{

return MYINT(a.x % b.x);

}

// 11. Cho đoạn chương trình sau:

#include <iostream.h>

main(){

cout << "Hello, world.\n";

}

Hãy sửa lại chương trình trên, nhưng không sửa chữa gì hàm main, để chương trình có thể tạo ra kết xuất:

Entering the Hello program saying...

Hello, world.

Then exiting...

#include <iostream>

using *namespace* std;

*class* *general*

{

*public:*

    general(){

        cout << "Entering the Hello program saying...\n";

    }

    ~general(){

        cout << "Then exiting...";

    }

};

*general* A;

*int* main() {

    cout << "Hello, world.\n";

    return 0;

}