TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

Thực hành Cấu trúc dữ liệu và giải thuật

LAB01

|  |  |
| --- | --- |
| **Sinh viên thực hiện:** | Nguyễn Hoài Quân MSSV: 3124411249 |

Thành phố Hồ Chí Minh, tháng 02 năm 2025

# LAB01b – Con trỏ cho đối tượng và mảng 1 chiều

## Con trỏ cho đối tượng

**Bài 1.** Cho hệ số a, b, c của phương trình bậc hai. Viết chương trình giải phương trình bậc hai (dùng con trỏ).

Moi ban nhap he so a, b, c: 1 2 1

Phuong trinh co 1 nghiem kep x = -1.

**Ý tưởng:**

* Input: a, b, c
* Output: màn hình
* Ví dụ: a = 1, b = 2, c = 1
* Màn hình:

1. Mời bạn nhập hệ số a, b, c: 1 2 1
2. Phương trình có 1 nghiệm kép x = -1

* Thuật toán:
* B1. Nhập a, b, c
* B2. Tính delta
* B3. Xuất nghiệm phương trình tương ứng

**Mã nguồn: LAB01b - Bai1.cpp**

LAB01b-Bai1.cpp

#include <iostream>

#include <cmath> // sqrt

using namespace std;

// Function prototype

void giaiPhuongTrinh(double \*a, double \*b, double \*c);

int main() {

double a, b, c;

// Nhập hệ số a, b, c

cout << "Moi ban nhap he so a, b, c: ";

cin >> a >> b >> c;

// Gọi hàm giải phương trình

giaiPhuongTrinh(&a, &b, &c);

return 0;

}

// Function definition

void giaiPhuongTrinh(double \*a, double \*b, double \*c) {

double delta, x1, x2;

// Tính delta

delta = (\*b) \* (\*b) - 4 \* (\*a) \* (\*c);

if (delta > 0) {

// Hai nghiệm phân biệt

x1 = (-(\*b) + sqrt(delta)) / (2 \* (\*a));

x2 = (-(\*b) - sqrt(delta)) / (2 \* (\*a));

cout << "Phuong trinh co 2 nghiem phan biet x1 = " << x1 << " va x2 = " << x2 << endl;

} else if (delta == 0) {

// Nghiệm kép

x1 = -(\*b) / (2 \* (\*a));

cout << "Phuong trinh co 1 nghiem kep x = " << x1 << endl;

} else {

// Phương trình vô nghiệm

cout << "Phuong trinh vo nghiem" << endl;

}

}

## Con trỏ cho mảng một chiều

**Bài 2.** Cho dãy số nguyên a có n phần tử (n ≤ 100). Hãy kiểm tra các tính chất sau của dãy a:

* Tính chẵn lẻ (các phần tử xen kẽ nhau chẵn lẻ)
* Tính toàn chẵn (dãy số chứa toàn số chẵn)

Moi ban nhap so luong phan tu: 5

Phan tu 0: 2

Phan tu 1: 5

Phan tu 2: 6

Phan tu 3: 5

Phan tu 4: 2

+ Day co tinh chat chan le

+ Day khong co tinh chat toan chan

**Ý tưởng:**

* Input: dãy số nguyên a, số lượng phần tử n
* Output: các kết quả kiểm tra tính chất chẵn lẻ và toàn chẵn.
* Ví dụ: n = 5
  + Màn hình:

1. Mời bạn nhập số lượng phần tử: 5
2. Phần tử 0: 2
3. Phần tử 1: 5
4. Phần tử 2: 6
5. Phần tử 3: 5
6. Phần tử 4: 2
   * Kết quả:
     + Dãy có tính chất chẵn lẻ
     + Dãy không có tính chất toàn chẵn

* Thuật toán:
  + B1. Nhập số lượng phần tử n và dãy số a.
  + B2. Kiểm tra tính chất chẵn lẻ.
  + B3. Kiểm tra tính chất toàn chẵn.
  + B4. Xuất kết quả.

**Mã nguồn: LAB01b - Bai2.cpp**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Moi ban nhap so luong phan tu: ";

cin >> n;

// Sử dụng con trỏ để cấp phát động mảng

int\* a = new int[n]; // Mảng được cấp phát động bằng con trỏ

// Nhập các phần tử cho mảng

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

cout << "Phan tu " << i << ": ";

cin >> \*(a + i); // Dùng con trỏ để truy cập phần tử mảng

}

// Kiểm tra tính chẵn lẻ

bool chanLe = true;

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

if ((\*(a + i) % 2 == 0 && \*(a + i + 1) % 2 == 0) || (\*(a + i) % 2 != 0 && \*(a + i + 1) % 2 != 0)) {

chanLe = false;

break;

}

}

// Kiểm tra tính toàn chẵn

bool toanChan = true;

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

if (\*(a + i) % 2 != 0) {

toanChan = false;

break;

}

}

// In kết quả

if (chanLe) {

cout << "+ Day co tinh chat chan le" << endl;

} else {

cout << "+ Day khong co tinh chat chan le" << endl;

}

if (toanChan) {

cout << "+ Day co tinh chat toan chan" << endl;

} else {

cout << "+ Day khong co tinh chat toan chan" << endl;

}

// Giải phóng bộ nhớ đã cấp phát động

delete[] a;

return 0;

}

**Bài 3.** Cho dãy ký tự a có n phần tử (n ≤ 100). Hãy tạo dãy b chứa các ký tự nguyên âm của a và xuất b ra màn hình.

Moi ban nhap so luong phan tu: 5

Phan tu 0: a

Phan tu 1: b

Phan tu 2: c

Phan tu 3: d

Phan tu 4: e

Day b chua cac ky tu nguyen am: a e

**Ý tưởng:**

* Input: dãy ký tự a, số lượng phần tử n
* Output: dãy ký tự b chứa các ký tự nguyên âm.
* Ví dụ: n = 5
  + Màn hình:
    1. Mời bạn nhập số lượng phần tử: 5
    2. Phần tử 0: a
    3. Phần tử 1: b
    4. Phần tử 2: c
    5. Phần tử 3: d
    6. Phần tử 4: e
  + Kết quả: Dãy b chứa các ký tự nguyên âm: a e
* Thuật toán:
  + B1. Nhập số lượng phần tử n và dãy ký tự a.
  + B2. Duyệt qua dãy a và tìm các ký tự nguyên âm.
  + B3. Tạo dãy b chứa các ký tự nguyên âm.
  + B4. Xuất dãy b ra màn hình.

**Mã nguồn:** **LAB01b - Bai3.cpp**

#include <iostream>

using namespace std;

bool isVowel(char c) {

// Kiểm tra nếu ký tự là nguyên âm (a, e, i, o, u)

c = tolower(c); // Chuyển ký tự về dạng thường để so sánh dễ dàng

return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');

}

int main() {

int n;

cout << "Moi ban nhap so luong phan tu: ";

cin >> n;

// Sử dụng con trỏ để cấp phát động mảng ký tự

char\* a = new char[n]; // Mảng ký tự a

char\* b = new char[n]; // Mảng ký tự b chứa các nguyên âm

// Nhập các ký tự vào dãy a

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

cout << "Phan tu " << i << ": ";

cin >> \*(a + i); // Sử dụng con trỏ để truy cập phần tử mảng

}

// Duyệt qua dãy a và tìm các ký tự nguyên âm

int j = 0; // Chỉ số cho dãy b

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

if (isVowel(\*(a + i))) { // Kiểm tra nếu ký tự a[i] là nguyên âm

\*(b + j) = \*(a + i); // Lưu ký tự nguyên âm vào dãy b

++j; // Chuyển đến vị trí tiếp theo trong dãy b

}

}

// In dãy b ra màn hình

cout << "Day cac ky tu nguyen am: ";

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

cout << \*(b + i) << " "; // In ra dãy ký tự nguyên âm

}

cout << endl;

// Giải phóng bộ nhớ đã cấp phát động

delete[] a;

delete[] b;

return 0;

}

**Bài 4.** Cho dãy số thực a có n phần tử và b có m phần tử (n, m ≤ 100) tăng dần. Hãy tạo dãy c từ hai dãy a, b sao cho tăng dần (không dùng sắp xếp).

+ Day so a

Moi ban nhap so luong phan tu: 3

Phan tu 0: 1.1

Phan tu 1: 3.2

Phan tu 2: 5.3

+ Day so b

Moi ban nhap so luong phan tu: 4

Phan tu 0: 2.1

Phan tu 1: 3.2

Phan tu 2: 5.4

Phan tu 3: 8.4

+ Day so c

Day so co 7 phan tu: 1.1 2.1 3.2 3.2 5.4 5.4 8.4

**Ý tưởng:**

* Input: dãy số thực a có n phần tử, dãy số thực b có m phần tử.
* Output: dãy số thực c chứa tất cả phần tử từ a và b, vẫn tăng dần.
* Thuật toán:
  + B1. Nhập số lượng phần tử của dãy a và b.
  + B2. Nhập dãy a và b theo thứ tự tăng dần.
  + B3. Duyệt cả hai dãy bằng con trỏ để tạo dãy c mà vẫn giữ nguyên thứ tự.
  + B4. Xuất dãy c ra màn hình.

**Mã nguồn: LAB01b - Bai4.cpp**

#include <iostream>

using namespace std;

// funtion prototype

void mergeSortedArrays(double\* a, int n, double\* b, int m, double\* c, int& sizeC);

int main() {

    int n, m;

    // Nhập dãy số a

    cout << "+ Day so a\nMoi ban nhap so luong phan tu: ";

    cin >> n;

    double\* a = new double[n]; *// Cấp phát động mảng a*

    for (int i = 0; i < n; ++i) {

        cout << "Phan tu " << i << ": ";

        cin >> \*(a + i);

    }

    // Nhập dãy số b

    cout << "+ Day so b\nMoi ban nhap so luong phan tu: ";

    cin >> m;

    double\* b = new double[m]; *// Cấp phát động mảng b*

    for (int i = 0; i < m; ++i) {

        cout << "Phan tu " << i << ": ";

        cin >> \*(b + i);

    }

    // Cấp phát động mảng c với kích thước tối đa n + m

    double\* c = new double[n + m];

    int sizeC;  // Số phần tử thực tế của dãy c

    // Gộp hai dãy số đã sắp xếp

    mergeSortedArrays(a, n, b, m, c, sizeC);

    // Xuất kết quả

    cout << "+ Day so c\nDay so co " << sizeC << " phan tu: ";

    for (int i = 0; i < sizeC; ++i) {

        cout << \*(c + i) << " ";

    }

    cout << endl;

    // Giải phóng bộ nhớ đã cấp phát động

    delete[] a;

    delete[] b;

    delete[] c;

    return 0;

}

// function definition

void mergeSortedArrays(double\* a, int n, double\* b, int m, double\* c, int& sizeC) {

    int i = 0, j = 0, k = 0;

    // Trộn hai dãy a và b vào dãy c theo thứ tự tăng dần

    while (i < n && j < m) {

        if (\*(a + i) <= \*(b + j)) {

            \*(c + k) = \*(a + i);

            i++;

        } else {

            \*(c + k) = \*(b + j);

            j++;

        }

        k++;

    }

    // Nếu còn phần tử trong dãy a

    while (i < n) {

        \*(c + k) = \*(a + i);

        i++;

        k++;

    }

    // Nếu còn phần tử trong dãy b

    while (j < m) {

        \*(c + k) = \*(b + j);

        j++;

        k++;

    }

    sizeC = k;  // Số phần tử thực tế của dãy c

}

**Bài 5.** Cho dãy số nguyên a có n phần tử (n ≤ 100). Hãy tách dãy a thành dãy b chứa số chẵn và dãy c chứa số lẻ.

+ Day so a

Moi ban nhap so luong phan tu: 3

Phan tu 0: 2

Phan tu 1: 5

Phan tu 2: 8

+ Day so b chua so chan

Day so co 2 phan tu: 2 8

+ Day so c

Day so co 1 phan tu: 5

**Ý tưởng:**

* Input: dãy số nguyên a, số lượng phần tử n.
* Output: dãy số b (số chẵn), dãy số c (số lẻ).
* Thuật toán:
  + B1. Nhập số lượng phần tử của dãy a.
  + B2. Phân loại từng phần tử vào dãy b (chẵn) hoặc dãy c (lẻ).
  + B3. Xuất kết quả.

**Mã nguồn: LAB01b - Bai5.cpp**

#include <iostream>

using namespace std;

// function prototype

void separateEvenOdd(int\* a, int n, int\* b, int& sizeB, int\* c, int& sizeC);

int main() {

int n;

// Nhập dãy số a

cout << "+ Day so a\nMoi ban nhap so luong phan tu: ";

cin >> n;

int\* a = new int[n]; // Cấp phát động mảng a

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

cout << "Phan tu " << i << ": ";

cin >> \*(a + i);

}

// Cấp phát động mảng b và c với kích thước tối đa n

int\* b = new int[n];

int\* c = new int[n];

int sizeB, sizeC;

// Phân tách dãy số a

separateEvenOdd(a, n, b, sizeB, c, sizeC);

// Xuất dãy số chẵn

cout << "+ Day so b chua so chan\n";

cout << "Day so co " << sizeB << " phan tu: ";

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

cout << \*(b + i) << " ";

}

cout << endl;

// Xuất dãy số lẻ

cout << "+ Day so c chua so le\n";

cout << "Day so co " << sizeC << " phan tu: ";

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

cout << \*(c + i) << " ";

}

cout << endl;

// Giải phóng bộ nhớ đã cấp phát động

delete[] a;

delete[] b;

delete[] c;

return 0;

}

//function definition

void separateEvenOdd(int\* a, int n, int\* b, int& sizeB, int\* c, int& sizeC) {

sizeB = 0;

sizeC = 0;

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

if (\*(a + i) % 2 == 0) { // Số chẵn

\*(b + sizeB) = \*(a + i);

sizeB++;

} else { // Số lẻ

\*(c + sizeC) = \*(a + i);

sizeC++;

}

}

}

# LAB01c – Cấu trúc

**Bài 1.** Một phòng ban có tối đa 100 nhân viên, mỗi nhân viên (**NhanVien**) gồm các thông tin sau: mã số (**MaSo**) – chuỗi tối đa 10 ký tự, họ lót (**Ho**) – chuỗi tối đa 10 ký tự, tên (**Ten**) – chuỗi tối đa 50 ký tự, Phái (**Phai**) – số nguyên có giá trị 0 cho Nữ và 1 cho Nam, số năm làm việc (**ThamNien**) – số nguyên dương lớn hơn hay bằng 0. Phòng ban (**PhongBan**) bao gồm mảng tối đa 100 nhân viên (**aNhanVien**) và số lượng nhân viên hiện có (**SoLuong**). Hãy cho biết sĩ số nam, nữ và in ra danh sách nhân viên tăng dần theo thâm niên.

**Yêu cầu:** Viết chương trình thực hiện yêu cầu đề bài, gồm các phần sau:

a) Khai báo cấu trúc NhanVien và PhongBan với thông tin mô tả phía trên.

b) Hàm nhập danh sách các nhân viên cho phòng ban có ràng buộc dữ liệu.

**void NhapPhongBan(PhongBan &pb) {...}**

c) Hàm xuất thông tin phòng ban ra màn hình.

**void XuatPhongBan(PhongBan pb) {...}**

d) Hàm đếm sĩ số nhân viên trong phòng ban.

**void DemSiSo(PhongBan ds, int &sonam, int &sonu) {...}**

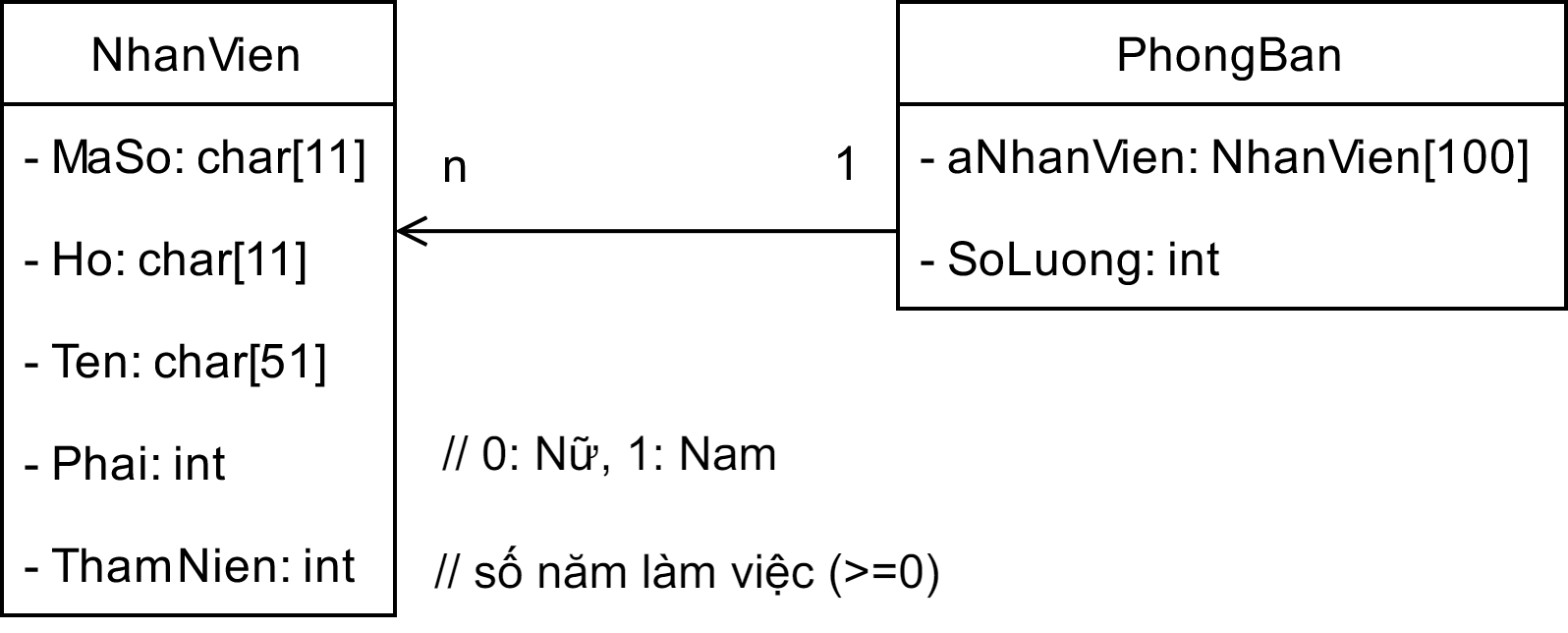
e) Hàm sắp xếp danh sách nhân viên tăng dần theo thâm niên.

**void SapXepTangTheoThamNien(PhongBan &pb) {...}**

f) Hàm main sử dụng các hàm trên xử lý các yêu cầu đề bài.

**int main() {...}**

**Sơ đồ lớp đối tượng:**



**Sơ đồ phối hợp hàm:**

main()

|

+--> NhapPhongBan(pb)

| |

| +--> NhapNhanVien(nv) (trong vòng lặp nhập từng nhân viên)

|

+--> DemSiSo(pb, sonam, sonu)

|

+--> SapXepTangTheoThamNien(pb)

|

+--> XuatPhongBan(pb)

|

+--> XuatNhanVien(nv) (trong vòng lặp xuất từng nhân viên)

**Mã nguồn: LAB01c - Bai1.1.cpp**

#include <iostream>

using namespace std;

// Khai báo cấu trúc NhanVien

struct NhanVien {

    char MaSo[11];     // Mã số tối đa 10 ký tự + ký tự kết thúc '\0'

    char Ho[11];       // Họ lót tối đa 10 ký tự

    char Ten[51];      // Tên tối đa 50 ký tự

    int Phai;          // 0: Nữ, 1: Nam

    int ThamNien;      // Số năm làm việc (>=0)

};

// Khai báo cấu trúc PhongBan

struct PhongBan {

    NhanVien aNhanVien[100];  // Mảng chứa tối đa 100 nhân viên

    int SoLuong;              // Số lượng nhân viên hiện có

};

// Hàm nhập thông tin 1 nhân viên

void NhapNhanVien(NhanVien &nv) {

    cout << "Nhap ma so: ";

    cin >> nv.MaSo;

    cout << "Nhap ho: ";

    cin >> nv.Ho;

    cout << "Nhap ten: ";

    cin >> nv.Ten;

    cout << "Nhap phai (0: Nu, 1: Nam): ";

    cin >> nv.Phai;

    while(nv.Phai != 0 && nv.Phai != 1) {

        cout << "Gia tri phai khong hop le. Nhap lai (0: Nu, 1: Nam): ";

        cin >> nv.Phai;

    }

    cout << "Nhap so nam lam viec (Tham Nien): ";

    cin >> nv.ThamNien;

    while(nv.ThamNien < 0) {

        cout << "So nam lam viec phai lon hon hoac bang 0. Nhap lai: ";

        cin >> nv.ThamNien;

    }

}

// b) Hàm nhập danh sách các nhân viên cho phòng ban

void NhapPhongBan(PhongBan &pb) {

    cout << "Nhap so luong nhan vien (0 - 100): ";

    cin >> pb.SoLuong;

    while(pb.SoLuong < 0 || pb.SoLuong > 100) {

        cout << "So luong nhan vien phai tu 0 den 100. Nhap lai: ";

        cin >> pb.SoLuong;

    }

    for(int i = 0; i < pb.SoLuong; i++) {

        cout << "\nNhap thong tin nhan vien thu " << i+1 << ":\n";

        NhapNhanVien(pb.aNhanVien[i]);

    }

}

// Hàm xuất thông tin của 1 nhân viên

void XuatNhanVien(NhanVien nv) {

    cout << nv.MaSo << "\t"

         << nv.Ho << "\t"

         << nv.Ten << "\t"

         << (nv.Phai == 1 ? "Nam" : "Nu") << "\t"

         << nv.ThamNien << endl;

}

// c) Hàm xuất thông tin phòng ban ra màn hình

void XuatPhongBan(PhongBan pb) {

    cout << "\nDanh sach nhan vien trong phong ban:\n";

    cout << "MaSo\tHo\tTen\tPhai\tThamNien\n";

    for(int i = 0; i < pb.SoLuong; i++) {

        XuatNhanVien(pb.aNhanVien[i]);

    }

}

// d) Hàm đếm sĩ số nhân viên trong phòng ban

void DemSiSo(PhongBan ds, int &sonam, int &sonu) {

    sonam = 0;

    sonu = 0;

    for(int i = 0; i < ds.SoLuong; i++) {

        if(ds.aNhanVien[i].Phai == 1)

            sonam++;

        else

            sonu++;

    }

}

// e) Hàm sắp xếp danh sách nhân viên tăng dần theo thâm niên

void SapXepTangTheoThamNien(PhongBan &pb) {

    // Sử dụng thuật toán sắp xếp nổi bọt (bubble sort)

    for(int i = 0; i < pb.SoLuong - 1; i++) {

        for(int j = i + 1; j < pb.SoLuong; j++) {

            if(pb.aNhanVien[i].ThamNien > pb.aNhanVien[j].ThamNien) {

                // Hoán đổi nhân viên

                NhanVien temp = pb.aNhanVien[i];

                pb.aNhanVien[i] = pb.aNhanVien[j];

                pb.aNhanVien[j] = temp;

            }

        }

    }

}

// f) Hàm main sử dụng các hàm trên xử lý các yêu cầu đề bài

int main() {

    PhongBan pb;

    // Nhập thông tin phòng ban (danh sách nhân viên)

    NhapPhongBan(pb);

    // Đếm sĩ số: số lượng nam và nữ

    int soNam, soNu;

    DemSiSo(pb, soNam, soNu);

    cout << "\nSi so nhan vien: Nam = " << soNam << ", Nu = " << soNu << endl;

    // Sắp xếp danh sách nhân viên tăng dần theo thâm niên

    SapXepTangTheoThamNien(pb);

    // Xuất danh sách nhân viên sau khi sắp xếp

    cout << "\nDanh sach nhan vien sau khi sap xep tang theo tham nien:\n";

    XuatPhongBan(pb);

    return 0;

}

**Bài 2.** Một cửa hàng bán hoa có tối đa 20 loại hoa, mỗi loại hoa gồm các thông tin sau: Tên loại (**Ten**), Số lượng (**SoLuong**), Đơn vị tính (**DVT**), Đơn giá (**DonGia**). Với tên loại, số lượng mà khách hàng yêu cầu, hãy cho biết loại đó có hay không. Nếu có, có đủ số lượng bán hay không và tổng tiền nếu đủ số lượng bán.

**Yêu cầu:** Viết chương trình thực hiện yêu cầu đề bài, gồm các phần sau:

a) Khai báo cấu trúc **LoaiHoa** và **DanhSachLoaiHoa** với thông tin mô tả phía trên.

b) Hàm nhập danh sách các loại hoa.

**void NhapDanhSach(DanhSachLoaiHoa &ds) {...}**

c) Hàm xuất danh sách ra màn hình.

**void XuatDanhSach(DanhSachLoaiHoa ds) {...}**

d) Hàm tìm loại hoa theo tên loại.

**int TimLoaiHoa(DanhSachLoaiHoa ds, char \*tenloai) {...}**

Trả về vị trí loại hoa trong danh sách, nếu không có trả về -1.

e) Hàm xử lý bán hoa cho khách hàng biết trước tên loại và số lượng.

**void XuLyBanHoa(DanhSachLoaiHoa &ds,char \*tenloai, int soluong) {...}**

**Chức năng:** Kiểm tra hoa có tồn tại không. Nếu có, kiểm tra xem số lượng có đủ không. Nếu đủ, in giá tiền, biết giá tiền bằng số lượng khách hàng mua \* đơn giá.

f) Hàm main sử dụng các hàm trên xử lý các yêu cầu đề bài.

**void main() {...}**

**Sơ đồ lớp đối tượng:**



**Sơ đồ phối hợp hàm:**

main()

|

+--> NhapDanhSach(ds)

| |

| +--> Nhập thông tin từng loại hoa

|

+--> XuatDanhSach(ds)

|

+--> Nhập tên loại hoa và số lượng khách hàng yêu cầu

|

+--> XuLyBanHoa(ds, tenloai, soluong)

|

+--> TimLoaiHoa(ds, tenloai)

**Mã nguồn: LAB01c - Bai1.2.cpp**

#include <iostream>

#include <cstring>

using namespace std;

// a) Khai báo cấu trúc LoaiHoa và DanhSachLoaiHoa

struct LoaiHoa {

    char Ten[50];      // Tên loại hoa

    int SoLuong;       // Số lượng hiện có

    char DVT[20];      // Đơn vị tính

    double DonGia; *// Đơn giá*

};

struct DanhSachLoaiHoa {

    LoaiHoa aLoaiHoa[20]; // Mảng chứa tối đa 20 loại hoa

    int SoLuong;         // Số lượng loại hoa hiện có

};

// b) Hàm nhập danh sách các loại hoa

void NhapDanhSach(DanhSachLoaiHoa &ds) {

    cout << "Nhap so luong loai hoa (toi da 20): ";

    cin >> ds.SoLuong;

    while(ds.SoLuong < 0 || ds.SoLuong > 20) {

        cout << "So luong loai hoa khong hop le. Nhap lai: ";

        cin >> ds.SoLuong;

    }

    cin.ignore(); // Xóa bộ đệm sau khi nhập số lượng

    for (int i = 0; i < ds.SoLuong; i++) {

        cout << "\nNhap thong tin loai hoa thu " << i + 1 << ":\n";

        cout << "Nhap ten loai hoa: ";

        cin.getline(ds.aLoaiHoa[i].Ten, 50);

        cout << "Nhap so luong: ";

        cin >> ds.aLoaiHoa[i].SoLuong;

        cin.ignore(); // Xóa bộ đệm sau khi nhập số

        cout << "Nhap don vi tinh: ";

        cin.getline(ds.aLoaiHoa[i].DVT, 20);

        cout << "Nhap don gia: ";

        cin >> ds.aLoaiHoa[i].DonGia;

        cin.ignore(); // Xóa bộ đệm sau khi nhập số

    }

}

// c) Hàm xuất danh sách ra màn hình

void XuatDanhSach(DanhSachLoaiHoa ds) {

    cout << "\nDanh sach cac loai hoa:\n";

    cout << "Ten loai hoa\tSo luong\tDon vi tinh\tDon gia\n";

    for (int i = 0; i < ds.SoLuong; i++) {

        cout << ds.aLoaiHoa[i].Ten << "\t\t"

             << ds.aLoaiHoa[i].SoLuong << "\t\t"

             << ds.aLoaiHoa[i].DVT << "\t\t"

             << ds.aLoaiHoa[i].DonGia << "\n";

    }

}

// d) Hàm tìm loại hoa theo tên loại

// Chuyển đổi chuỗi về chữ thường

void ChuyenChuThuong(char \*str) {

    for (int i = 0; str[i]; i++) {

        str[i] = tolower(str[i]);

    }

}

// Trả về vị trí trong danh sách nếu tìm thấy, nếu không trả về -1

int TimLoaiHoa(DanhSachLoaiHoa ds, char \*tenloai) {

    char tenTimKiem[50];

    strcpy(tenTimKiem, tenloai);

    ChuyenChuThuong(tenTimKiem);  // Chuyển tên cần tìm về chữ thường

    for (int i = 0; i < ds.SoLuong; i++) {

        char tenTrongDS[50];

        strcpy(tenTrongDS, ds.aLoaiHoa[i].Ten);

        ChuyenChuThuong(tenTrongDS); // Chuyển tên trong danh sách về chữ thường

        if (strcmp(tenTimKiem, tenTrongDS) == 0)

            return i;

    }

    return -1;

}

// e) Hàm xử lý bán hoa cho khách hàng theo tên loại và số lượng yêu cầu

void XuLyBanHoa(DanhSachLoaiHoa &ds, char \*tenloai, int soluong) {

    int pos = TimLoaiHoa(ds, tenloai);

    if (pos == -1) {

        cout << "\nLoai hoa '" << tenloai << "' khong ton tai trong danh sach.\n";

        return;

    }

    // Kiểm tra số lượng hiện có

    if (ds.aLoaiHoa[pos].SoLuong < soluong) {

        cout << "\nSo luong hoa hien co khong du de ban.\n";

        return;

    }

    // Nếu đủ số lượng, tính tổng tiền

    double tongTien = soluong \* ds.aLoaiHoa[pos].DonGia;

    cout << "\nTong tien can thanh toan: " << tongTien << "\n";

    // (Tùy chọn) Cập nhật lại số lượng hoa sau khi bán

    ds.aLoaiHoa[pos].SoLuong -= soluong;

}

// f) Hàm main sử dụng các hàm trên để xử lý yêu cầu

int main() {

    DanhSachLoaiHoa ds;

    // Nhập danh sách các loại hoa

    NhapDanhSach(ds);

    // Xuất danh sách các loại hoa

    XuatDanhSach(ds);

    // Nhập thông tin yêu cầu bán hoa của khách hàng

    char tenLoaiCanMua[50];

    int soLuongMua;

    cout << "\nNhap ten loai hoa can mua: ";

    cin.getline(tenLoaiCanMua, 50);

    cout << "Nhap so luong can mua: ";

    cin >> soLuongMua;

    // Xử lý bán hoa

    XuLyBanHoa(ds, tenLoaiCanMua, soLuongMua);

    return 0;

}

# LAB01c – Cấu trúc (bis)

**Bài tập 1. Quản lý điện thoại**

**Tên file chương trình:QuanLyDienThoai.\***

Thông tin một điện thoại gồm:

• Mã điện thoại (tối đa 10 ký tự)

• Nhãn hiệu (tối đa 20 ký tự)

• Giá (số nguyên)

Viết chương trình quản lý một danh sách n (n ≤ 100) điện thoại. Chương trình có những chức năng sau:

a) Thêm 1 điện thoại vào danh sách điện điện thoại. Nếu điện thoại đã tồn tại trong danh sách thì hay danh sách bị đầy thì thông báo lỗi.

b) Cho mã điện thoại, in nhãn hiệu và giá của điện thoại đó. Nếu mã điện thoại không có trong danh sách thì báo lỗi.

c) Cho mã điện thoại, cập nhật lại giá của điện thoại đó. Nếu mã điện thoại không có trong danh sách thì báo lỗi.

d) Cho mã điện thoại, xóa điện thoại đó. Nếu mã điện thoại không có trong danh sách thì báo lỗi.

e) Xuất tất cả điện thoại trong danh sách.

f) Xây dựng menu cho chương trình này.

**Mã nguồn: QuanLyDienThoai.cpp**

#include <iostream>

#include <cstring>

using namespace std;

const int MAX = 100; // Số lượng điện thoại tối đa

// Khai báo cấu trúc DienThoai

struct DienThoai {

char ma[11]; // Mã điện thoại (tối đa 10 ký tự)

char nhanHieu[21]; // Nhãn hiệu (tối đa 20 ký tự)

int gia; // Giá (số nguyên)

};

// Hàm tìm kiếm điện thoại theo mã

// Trả về vị trí trong mảng nếu tìm thấy, nếu không trả về -1.

int TimDienThoai(DienThoai ds[], int n, const char \*ma) {

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

if (strcmp(ds[i].ma, ma) == 0)

return i;

}

return -1;

}

// a) Hàm thêm 1 điện thoại vào danh sách

void ThemDienThoai(DienThoai ds[], int &n) {

if (n >= MAX) {

cout << "Danh sach da day, khong the them dien thoai moi!\n";

return;

}

DienThoai dt;

cout << "Nhap ma dien thoai (toi da 10 ky tu): ";

cin >> dt.ma;

// Kiểm tra xem mã điện thoại đã tồn tại hay chưa

if (TimDienThoai(ds, n, dt.ma) != -1) {

cout << "Dien thoai voi ma " << dt.ma << " da ton tai trong danh sach!\n";

return;

}

cout << "Nhap nhan hieu (toi da 20 ky tu): ";

cin.ignore(); // Xóa ký tự newline trong bộ đệm

cin.getline(dt.nhanHieu, 21);

cout << "Nhap gia: ";

cin >> dt.gia;

ds[n] = dt;

n++;

cout << "Them dien thoai thanh cong!\n";

}

// b) Hàm cho mã điện thoại, in nhãn hiệu và giá của điện thoại đó

void TimVaInThongTin(DienThoai ds[], int n) {

char ma[11];

cout << "Nhap ma dien thoai can tim: ";

cin >> ma;

int pos = TimDienThoai(ds, n, ma);

if (pos == -1) {

cout << "Khong tim thay dien thoai voi ma " << ma << "!\n";

} else {

cout << "\nThong tin dien thoai:\n";

cout << "Ma: " << ds[pos].ma << "\n";

cout << "Nhan hieu: " << ds[pos].nhanHieu << "\n";

cout << "Gia: " << ds[pos].gia << "\n";

}

}

// c) Hàm cập nhật lại giá của điện thoại theo mã

void CapNhatGia(DienThoai ds[], int n) {

char ma[11];

cout << "Nhap ma dien thoai can cap nhat gia: ";

cin >> ma;

int pos = TimDienThoai(ds, n, ma);

if (pos == -1) {

cout << "Khong tim thay dien thoai voi ma " << ma << "!\n";

} else {

cout << "Gia hien tai cua dien thoai: " << ds[pos].gia << "\n";

cout << "Nhap gia moi: ";

int giaMoi;

cin >> giaMoi;

ds[pos].gia = giaMoi;

cout << "Cap nhat gia thanh cong!\n";

}

}

// d) Hàm xóa điện thoại theo mã

void XoaDienThoai(DienThoai ds[], int &n) {

char ma[11];

cout << "Nhap ma dien thoai can xoa: ";

cin >> ma;

int pos = TimDienThoai(ds, n, ma);

if (pos == -1) {

cout << "Khong tim thay dien thoai voi ma " << ma << "!\n";

} else {

// Dịch chuyển các phần tử sau vị trí pos sang bên trái

for (int i = pos; i < n - 1; i++) {

ds[i] = ds[i + 1];

}

n--;

cout << "Xoa dien thoai thanh cong!\n";

}

}

// e) Hàm xuất tất cả điện thoại trong danh sách

void XuatDanhSach(DienThoai ds[], int n) {

if (n == 0) {

cout << "Danh sach rong!\n";

return;

}

cout << "\nDanh sach dien thoai:\n";

cout << "Ma\tNhan hieu\tGia\n";

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

cout << ds[i].ma << "\t" << ds[i].nhanHieu << "\t\t" << ds[i].gia << "\n";

}

}

// f) Hàm hiển thị menu

void Menu() {

cout << "\n======== MENU QUAN LY DIEN THOAI ========\n";

cout << "1. Them 1 dien thoai\n";

cout << "2. Tim va in thong tin dien thoai theo ma\n";

cout << "3. Cap nhat gia dien thoai\n";

cout << "4. Xoa dien thoai\n";

cout << "5. Xuat danh sach dien thoai\n";

cout << "6. Thoat chuong trinh\n";

cout << "=========================================\n";

cout << "Chon chuc nang: ";

}

int main() {

DienThoai ds[MAX];

int n = 0; // Số lượng điện thoại hiện có trong danh sách

int choice; // Lựa chọn của người dùng

do {

Menu();

cin >> choice;

switch (choice) {

case 1:

ThemDienThoai(ds, n);

break;

case 2:

TimVaInThongTin(ds, n);

break;

case 3:

CapNhatGia(ds, n);

break;

case 4:

XoaDienThoai(ds, n);

break;

case 5:

XuatDanhSach(ds, n);

break;

case 6:

cout << "Thoat chuong trinh. Tam biet!\n";

break;

default:

cout << "Chuc nang khong hop le. Vui long chon lai!\n";

}

} while (choice != 6);

return 0;

}

**Bài tập 2. Cửa hàng tạp hoá**

**Tên file chương trình:CuaHangTapHoa.\***

Một cửa hàng tạp hóa cần quản lý danh mục mặt hàng trong cửa hàng. Mỗi mặt hàng gồm các thông tin sau:

• MSMH (Mã số mặt hàng)

• TenMH (Tên mặt hàng)

• SoLuong (Số lượng mặt hàng)

• DonGia (Đơn giá)

Viết chương trình thực hiện các chức năng:

a) Nhập n mặt hàng (n ≤ 100)

b) Cho tên mặt hàng, hãy in ra thông tin đầy đủ của mặt hàng có tên đó (Nếu không tìm thấy thì thông báo “Khong co mat hang dang tim”)

c) Tính tổng số lượng các mặt hàng

**Mã nguồn: CuaHangTapHoa.cpp**

#include <iostream>

#include <cstring>

using namespace std;

*// Khai báo cấu trúc MatHang*

struct MatHang {

    char msmh[20]; *// Mã số mặt hàng*

    char tenmh[50]; *// Tên mặt hàng*

    int soLuong; *// Số lượng mặt hàng*

    double donGia; *// Đơn giá*

};

*// a) Hàm nhập thông tin 1 mặt hàng*

void NhapMatHang(MatHang &mh) {

    cout << "Nhap MSMH (ma so mat hang): ";

    cin.getline(mh.msmh, 20);

    cout << "Nhap Ten mat hang: ";

    cin.getline(mh.tenmh, 50);

    cout << "Nhap So luong: ";

    cin >> mh.soLuong;

    cout << "Nhap Don gia: ";

    cin >> mh.donGia;

    cin.ignore(); *// Xóa ký tự newline sau khi nhập số*

}

*// a) Hàm nhập danh sách các mặt hàng (n <= 100)*

void NhapDanhSach(MatHang ds[], int &n) {

    cout << "Nhap so luong mat hang (n <= 100): ";

    cin >> n;

    while(n < 0 || n > 100) {

        cout << "So luong khong hop le. Nhap lai (n <= 100): ";

        cin >> n;

    }

    cin.ignore(); *// xóa newline*

    for (int i = 0; i < n; i++) {

        cout << "\nNhap thong tin mat hang thu " << i + 1 << ":\n";

        NhapMatHang(ds[i]);

    }

}

*// Hàm xuất thông tin của 1 mặt hàng*

void XuatMatHang(const MatHang &mh) {

    cout << "MSMH: " << mh.msmh << "\n";

    cout << "Ten mat hang: " << mh.tenmh << "\n";

    cout << "So luong: " << mh.soLuong << "\n";

    cout << "Don gia: " << mh.donGia << "\n";

}

*// b) Hàm tìm mặt hàng theo tên (so sánh chuỗi TenMH)*

*// Trả về vị trí trong mảng nếu tìm thấy, nếu không trả về -1.*

int TimMatHang(MatHang ds[], int n, const char \*ten) {

    for (int i = 0; i < n; i++) {

        if (strcmp(ds[i].tenmh, ten) == 0) {

            return i;

        }

    }

    return -1;

}

*// c) Hàm tính tổng số lượng của các mặt hàng trong danh sách*

int TinhTongSoLuong(MatHang ds[], int n) {

    int tong = 0;

    for (int i = 0; i < n; i++) {

        tong += ds[i].soLuong;

    }

    return tong;

}

int main() {

    MatHang ds[100];

    int n;

*// a) Nhập danh sách mặt hàng*

    NhapDanhSach(ds, n);

*// b) Cho tên mặt hàng, tìm và in ra thông tin*

    char tenTim[50];

    cout << "\nNhap ten mat hang can tim: ";

    cin.getline(tenTim, 50);

    int pos = TimMatHang(ds, n, tenTim);

    if (pos == -1) {

        cout << "Khong co mat hang dang tim\n";

    } else {

        cout << "\nThong tin cua mat hang:\n";

        XuatMatHang(ds[pos]);

    }

*// c) Tính tổng số lượng các mặt hàng và in ra*

    int tongSoLuong = TinhTongSoLuong(ds, n);

    cout << "\nTong so luong cac mat hang: " << tongSoLuong << "\n";

    return 0;

}

# LAB01d – Xử lí chuỗi

**Bài 1.** Cho chuỗi (|s| ≤ 1000). Hãy sắp các ký tự trong chuỗi theo thứ tự tăng dần bằng C++.

**Ví dụ:**

Moi ban nhap chuoi s: abcba

Chuoi “abcba” sau khi sap xep: aabbc

**Mã nguồn: LAB01d - Bai1.cpp**

#include <iostream>

#include <algorithm>

#include <string>

using namespace std;

int main() {

string s;

cout << "Moi ban nhap chuoi s: ";

getline(cin, s); // Nhập cả dòng (để bao gồm cả khoảng trắng nếu có)

// Sắp xếp các ký tự trong chuỗi theo thứ tự tăng dần (theo mã ASCII)

sort(s.begin(), s.end());

cout << "Chuoi \"" << s << "\" sau khi sap xep: " << s << endl;

return 0;

}

**Bài 2.** Cho chuỗi (|s| ≤ 1000) và số nguyên k (0 ≤ k ≤ |s|-1). Hãy viết chương trình xóa ký tự tại vị trí thứ k trong chuỗi.

**Ví dụ:**

Moi ban nhap chuoi s: abcba

Moi ban nhap vi tri can xoa: 1

Chuoi “abcba” sau khi xoa ky tu tai vi tri 1: acba

**Mã nguồn: LAB01d - Bai2.cpp**

#include <iostream>

#include <string>

using namespace std;

int main() {

    string s;

    cout << "Moi ban nhap chuoi s: ";

    getline(cin, s); *// Đọc cả dòng để lấy chuỗi, kể cả khoảng trắng nếu có*

    int k;

    cout << "Moi ban nhap vi tri can xoa: ";

    cin >> k;

*// Kiểm tra xem vị trí k có hợp lệ không (0 <= k < độ dài chuỗi)*

    if (k < 0 || k >= s.size()) {

        cout << "Vi tri " << k << " khong hop le!" << endl;

        return 1;

    }

*// Xóa ký tự tại vị trí k*

    s.erase(k, 1);

    cout << "Chuoi \"" << s << "\" sau khi xoa ky tu tai vi tri " << k << ": " << s << endl;

    return 0;

}

**Bài 3.** Cho chuỗi (|s| ≤ 1000), số nguyên k (0 ≤ k ≤ |s|-1) và kí tự c. Viết hàm chèn ký tự c vào chuỗi s tại vị trí k.

**Ví dụ:**

Moi ban nhap chuoi s: abcba

Moi ban nhap vi tri can chen: 1

Moi ban nhap ky tu chen: d

Chuoi “abcba” sau khi them ky tu “d” vao vi tri 1: adbcba

**Mã nguồn: LAB01d - Bai3.cpp**

#include <iostream>

#include <string>

using namespace std;

int main() {

    string s;

    cout << "Moi ban nhap chuoi s: ";

    getline(cin, s); *// Nhập cả dòng, bao gồm khoảng trắng nếu có*

    int k;

    cout << "Moi ban nhap vi tri can chen: ";

    cin >> k;

*// Kiểm tra tính hợp lệ của vị trí k (0 <= k <= |s|-1)*

    if (k < 0 || k >= s.size()) {

        cout << "Vi tri chen khong hop le!" << endl;

        return 1;

    }

    char c;

    cout << "Moi ban nhap ky tu chen: ";

    cin >> c;

*// Chèn ký tự c vào vị trí k trong chuỗi s*

    s.insert(s.begin() + k, c);

    cout << "Chuoi \"" << s << "\" sau khi them ky tu \"" << c

         << "\" vao vi tri " << k << ": " << s << endl;

    return 0;

}

# LAB01e – File văn bản

**Bài 1. Đọc các số nguyên**

**Tên file chương trình: DocCacSoNguyen.\***

Viết hàm đọc file chứa n số nguyên. Xuất kết quả đọc được ra màn hình.

**Dữ liệu vào:** Từ file văn bản **DaySoNguyen.inp** gồm

• Dòng thứ 1: Chứa số nguyên n là số phần tử của mảng.

• Dòng thứ 2: Chứa n số nguyên (các số cách nhau ít nhất một khoảng trắng).

**Ví dụ:** file **DaySoNguyen.inp**

5

4 3 5 3 2

**Mã nguồn: DocCacSoNguyen.cpp**

#include <iostream>

#include <fstream>

using namespace std;

// Hàm đọc file chứa n số nguyên và xuất ra màn hình

void docCacSoNguyen(const char\* tenFile) {

ifstream inFile(tenFile); // Mở file để đọc

if (!inFile) {

cerr << "Lỗi: Không thể mở file " << tenFile << endl;

return;

}

int n;

inFile >> n; // Đọc số lượng phần tử từ dòng đầu tiên

// Cấp phát mảng động chứa n số nguyên

int\* arr = new int[n];

// Đọc n số nguyên từ file (dòng thứ 2)

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

inFile >> arr[i];

}

inFile.close(); // Đóng file sau khi đọc xong

// Xuất kết quả ra màn hình

cout << "So luong phan tu: " << n << endl;

cout << "Cac so nguyen doc duoc: ";

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

cout << arr[i] << " ";

}

cout << endl;

// Giải phóng bộ nhớ đã cấp phát

delete[] arr;

}

int main() {

// Gọi hàm đọc file với tên file "DaySoNguyen.inp"

docCacSoNguyen("DaySoNguyen.inp");

return 0;

}

**Bài 2. Đọc mảng 2 chiều**

**Tên file chương trình: DocMang2CSoNguyen.\***

Viết hàm đọc file chứa mảng 2 chiều các số nguyên. Xuất kết quả đọc được ra màn hình.

**Dữ liệu vào:** Từ file văn bản **MangSo.inp** gồm

• Dòng thứ 1: Chứa 2 số nguyên n, m là số dòng và số cột của mảng.

• n dòng tiếp theo: Mỗi dòng chứa n số nguyên (các số cách nhau ít nhất một khoảng trắng)

**Ví dụ:** file **MangSo.inp**

3 5

5 3 5 7 2

4 5 5 7 1

5 4 5 3 9

**Mã nguồn: DocMang2CSoNguyen.cpp**

#include <iostream>

#include <fstream>

using namespace std;

// Hàm đọc file chứa mảng 2 chiều các số nguyên

void docMang2CSoNguyen(const char\* tenFile) {

ifstream inFile(tenFile); // Mở file để đọc

if (!inFile) {

cerr << "Loi: Khong the mo file " << tenFile << endl;

return;

}

int n, m;

inFile >> n >> m; // Đọc số dòng và số cột từ dòng đầu tiên của file

// Cấp phát mảng 2 chiều động với n dòng và m cột

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

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

arr[i] = new int[m];

}

// Đọc dữ liệu từ file cho từng phần tử của mảng

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

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

inFile >> arr[i][j];

}

}

inFile.close(); // Đóng file sau khi đọc xong

// Xuất mảng ra màn hình

cout << "Ma tran " << n << " x " << m << " doc duoc tu file:" << endl;

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

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

cout << arr[i][j] << "\t";

}

cout << endl;

}

// Giải phóng bộ nhớ đã cấp phát cho mảng 2 chiều

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

delete[] arr[i];

}

delete[] arr;

}

int main() {

// Gọi hàm với tên file "MangSo.inp"

docMang2CSoNguyen("MangSo.inp");

return 0;

}

**Bài 3. Tìm số nguyên tố**

**Tên file chương trình: NguyenTo.\***

Cho dãy n số nguyên . Hãy liệt kê các số nguyên tố trong dãy theo thứ tự tăng dần.

**Dữ liệu vào:** Từ file văn bản **NT.INP** gồm có

- Dòng đầu là n (1 < n ≤ 10000)

- n dòng tiếp theo, dòng thứ i ghi số

**Kết quả:** Ghi ra file văn bản NT.OUT

- Dòng đầu ghi số m là số lượng số nguyên tố (Nếu không có số nguyên tố nào thì ghi số 0)

- Dòng thứ 2 ghi m số nguyen tố tìm được (các số cách nhau ít nhất một khoảng trắng)

**Ví dụ:**

file **NT.INP**

6

7

5

6

1

19

4

file **NT.OUT**

3

5 7 19

**Mã nguồn: NguyenTo.cpp**

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

#include <cmath>

using namespace std;

// Hàm kiểm tra số nguyên tố

bool isPrime(int n) {

if (n <= 1)

return false; // Số nhỏ hơn hoặc bằng 1 không phải số nguyên tố

if (n == 2)

return true; // 2 là số nguyên tố duy nhất chẵn

if (n % 2 == 0)

return false;

int limit = static\_cast<int>(sqrt(n));

for (int i = 3; i <= limit; i += 2) {

if (n % i == 0)

return false;

}

return true;

}

int main() {

// Mở file NT.INP để đọc dữ liệu

ifstream fin("NT.INP");

if (!fin) {

cerr << "Loi: Khong the mo file NT.INP" << endl;

return 1;

}

int n;

fin >> n; // Đọc số lượng phần tử

vector<int> primeNumbers; // Vector lưu các số nguyên tố

// Đọc n số và kiểm tra số nguyên tố

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

int a;

fin >> a;

if (isPrime(a))

primeNumbers.push\_back(a);

}

fin.close(); // Đóng file sau khi đọc xong

// Sắp xếp các số nguyên tố theo thứ tự tăng dần

sort(primeNumbers.begin(), primeNumbers.end());

// Mở file NT.OUT để ghi kết quả

ofstream fout("NT.OUT");

if (!fout) {

cerr << "Loi: Khong the mo file NT.OUT" << endl;

return 1;

}

// Ghi số lượng số nguyên tố tìm được

fout << primeNumbers.size() << endl;

// Ghi các số nguyên tố, cách nhau bằng một khoảng trắng

for (size\_t i = 0; i < primeNumbers.size(); i++) {

fout << primeNumbers[i];

if (i != primeNumbers.size() - 1)

fout << " ";

}

fout << endl;

fout.close(); // Đóng file NT.OUT

return 0;

}

**Bài 4. Tìm cặp số**

**Tên file chương trình: CapSo.\***

Cho dãy n số nguyên và số nguyên k. Hãy tìm các cặp sao cho .

**Dữ liệu vào:** Từ file văn bản CapSo.INP gồm có

- Dòng đầu chứa số n (1 < n ≤ 10000) và k

- n dòng tiếp theo, dòng thứ i ghi số

**Kết quả:** Ghi ra file văn bản **CapSo.OUT**

- Nếu không tìm được cặp số nào ghi số 0. Ngược lại ghi các cặp số tìm được, mỗi cặp nằm trên 1 dòng (Các số cách nhau ít nhất 1 khoảng trắng).

**Ví dụ:**

file **CapSo.INP**

6 5

2

5

3

1

4

1

file **CapSo.OUT**

2 3

1 4

4 1

**Mã nguồn: NguyenTo.cpp**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

// Mở file CapSo.INP để đọc dữ liệu

ifstream fin("CapSo.INP");

if (!fin) {

cerr << "Loi: Khong the mo file CapSo.INP" << endl;

return 1;

}

int n, k;

fin >> n >> k; // Đọc số lượng phần tử và số k

// Cấp phát mảng động lưu n số nguyên

int\* arr = new int[n];

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

fin >> arr[i];

}

fin.close(); // Đóng file sau khi đọc xong

// Đếm số cặp thoả mãn: a\_i + a\_j = k với i < j

int count = 0;

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

for (int j = i + 1; j < n; j++) {

if (arr[i] + arr[j] == k) {

count++;

}

}

}

// Mở file CapSo.OUT để ghi kết quả

ofstream fout("CapSo.OUT");

if (!fout) {

cerr << "Loi: Khong the mo file CapSo.OUT" << endl;

delete[] arr; // Giải phóng bộ nhớ

return 1;

}

// Nếu không tìm được cặp số nào, ghi ra số 0

if (count == 0) {

fout << 0;

} else {

// Nếu có cặp, duyệt lại mảng để ghi các cặp thoả mãn vào file

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

for (int j = i + 1; j < n; j++) {

if (arr[i] + arr[j] == k) {

fout << arr[i] << " " << arr[j] << "\n";

}

}

}

}

fout.close(); // Đóng file kết quả

delete[] arr; // Giải phóng bộ nhớ đã cấp phát

return 0;

}