**câu 2: Trong các thuật toán sắp xếp, bạn thích nhất thuật toán nào? Thuật toán nào bạn không thích nhất? Tại sao?**

* Thuật toán thích nhất là Quicksort: Vì Quicksort So sánh từng cặp phần tử liền kề và hoán đổi nếu chúng không theo đúng thứ tự. Lặp lại quá trình cho đến khi danh sách được sắp xếp. Quicksort có độ phức tạp trung bình O(n log n), chạy rất nhanh trên hầu hết các bộ dữ liệu thực tế, và sử dụng phân hoạch tối ưu hóa việc sắp xếp. Dễ cài đặt dễ hiểu, Hiệu suất tốt ngay cả với dữ liệu lớn
* Thuật toán không thích nhất là Bubble Sort: VÌ Bubble Sort so sánh từng cặp phần tử liền kề và hoán đổi nếu chúng không theo đúng thức tự. Lặp lại quá trình cho đến khi danh sách được sắp xếp và hiệu suất kém trên các bộ dữ liệu lớn và ít được sử dụng trong thực tế. Nó chỉ hữu ích cho mục đích giảng dạy hoặc trong các tình huống đặc biệt khi dữ liệu gần như đã được sắp xếp.

**Câu 4/ Bài tập cơ sở**

* **Interchange Sort (sắp xếp đổi chỗ trực tiếp)**

+ Ý tưởng: So sánh từng phần tử với tất cả các phần tử phía sau nó. Nếu tìm thấy phần tử nhỏ hơ, đổi chỗ 2 phần tử.

+ Vd: Cho mảng [ 5, 3, 8, 4, 2], quá trình thực hiện

1. 5 so sánh với 3 đổi chỗ => [3, 5, 8, 4, 2]
2. 3 so sánh với 8 không đổi
3. 3 so sánh với 4 không đổi
4. 3 so sánh với 2 đổi chỗ => [2, 5, 8, 4, 3]
5. Tiếp tục cho đến khi sắp xếp xong => [2, 3, 4, 5, 8]

+ Code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file docx

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Interchange Sort

void interchangeSort(std::vector<int> &arr) {

    int n = arr.size();

    for (int i = 0; i < n - 1; i++)

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

            if (arr[i] > arr[j]) std::swap(arr[i], arr[j]);

}

// Đo thời gian chạy Interchange Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    interchangeSort(data);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ Test:

In put: [9 4 2 5 4 2 5 3]

Out put: [2 2 3 4 4 5 5 9]

+ Thời gian thực hiên: 1ms đối với dữ liệu nhỏ. Dữ liệu lớn 30.000 là 5s

* **Selection sort (sắp xếp chọn trực tiếp)**

**+** Ý tưởng: Tìm phần tử nhỏ nhất và đặt nó về đầu danh sách.

**+** Vd cho mảng [5, 3, 8, 4, 2]

1. Tìm phần tử nhỏ nhất 2 đổi với phần tử đầu => [2, 3, 8, 4, 5]
2. Tìm phần tử nhỏ nhất trong phần còn lại là 3 giữ nguyên
3. Tìm phần tử nhỏ nhất 4 đổi với 8 => [2 3 4 8 5]
4. Tìm phần tử nhỏ nhất 5 đổi với 8 => [2 3 4 5 8]

+ Code:

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Selection Sort

void selectionSort(std::vector<int> &arr) {

    int n = arr.size();

    for (int i = 0; i < n - 1; i++) {

        int minIdx = i;

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

            if (arr[j] < arr[minIdx]) minIdx = j;

        }

        std::swap(arr[i], arr[minIdx]);

    }

}

// Đo thời gian chạy Selection Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    selectionSort(data);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ Test case:

In put: 9 4 2 5 4 2 5 3

Out put: 2 2 3 4 4 5 5 9

+ Thời gian thực hiện: 0m với dữ liệu nhỏ. Dữ liệu lớn 30.000 thì cần đến 7s

* **Intertion sort (sắp xếp chèn trực tiếp)**

+ Ý tưởng: chọn một phần tử đặt nó vào vị trí đúng trong dãy sắp xếp.

+ Vd cho mảng [ 5 3 8 4 2] quá trình thực hiện

1. 3 so sánh với 5, chèn vào trước => 3 5 8 4 2
2. 8 đã đúng vị trí
3. 4 chèn vào trước 8 => [ 3 4 5 8 2]
4. 2 chèn vào trước tất cả => [ 2 3 4 5 8]

+ Code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Insertion Sort

void insertionSort(std::vector<int> &arr) {

    int n = arr.size();

    for (int i = 1; i < n; i++) {

        int key = arr[i];

        int j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j--;

        }

        arr[j + 1] = key;

    }

}

// Đo thời gian chạy Insertion Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    insertionSort(data);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ Test case

Input: [ 9 4 2 5 4 2 5 3]

Out put [2 2 3 4 4 5 5 9]

+ Thời gian thực hiện: 0ms với dữ liệu nhỏ với dữ liệu lớn 30.000 thì thời gian thực hiên là 6s.

* Bubble sort (sắp xếp nổi bọt)

+ Ý tưởng: So sánh từng cặp phần tử liền kề nhau và đổi chỗ nếu sai thứ tự.

+ Vd cho mảng [5 3 8 4 2]. Quá trình thực hiện

1. So sánh 5 và 3 đổi chổ => [3 5 8 4 2]
2. So sánh 5 và 8 không đổi
3. So sánh 8 và 4 đổi chỗ =>[ 3 5 4 8 2]
4. So sánh 8 và 2 đổi chổ => [ 3 5 4 2 8]
5. Tiếp tục cho đến khi sắp xếp xong => [ 2 3 4 5 8]

+ Code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Bubble Sort

void bubbleSort(std::vector<int> &arr) {

    int n = arr.size();

    for (int i = 0; i < n - 1; i++) {

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

            if (arr[j] > arr[j + 1]) {

                std::swap(arr[j], arr[j + 1]);

            }

        }

    }

}

// Đo thời gian chạy Bubble Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    bubbleSort(data);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ Test

Input: 9 4 2 5 4 2 5 3

Out put: 2 2 3 4 4 5 5 9

+ Thời gian thực hiên với dữ liệu nhỏ là 0ms với dữ liệu lớn 30,000 là 15s

* **Quick sort (Sắp xếp nhanh)**

+ Ý tưởng: chọn một phần tử làm chốt (pivot), chia mảng thành 2 phần (bé hơn và lớn hơn pivot)

+ VD cho mảng [5 3 8 4 2] quá trình thực hiện

Pivot = 2 => [2] | [5 3 8 4]

Pivot =5 => [2 3 4 ] | [5 8]

Cuối cùng => [ 2 3 4 5 8]

+ code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Quick Sort

void quickSort(std::vector<int> &arr, int left, int right) {

    if (left >= right) return;

    int pivot = arr[right];

    int i = left - 1;

    for (int j = left; j < right; j++) {

        if (arr[j] < pivot) std::swap(arr[++i], arr[j]);

    }

    std::swap(arr[++i], arr[right]);

    quickSort(arr, left, i - 1);

    quickSort(arr, i + 1, right);

}

// Đo thời gian chạy Quick Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    quickSort(data, 0, data.size() - 1);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+Test

Input 9 4 2 5 4 2 5 3

Out put 2 2 3 4 4 5 5 9

+ Thời gian thực hiện đối với dữ liệu nhỏ là 0ms với dữ liệu lớn là 100ms

* **Merge sort ( sắp xếp trộn)**

+ Ý tưởng: chi mảng thành 2 phần, sắp xếp từng phần rồi trộn lại

+ code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Merge Sort

void merge(std::vector<int> &arr, int left, int mid, int right) {

    std::vector<int> temp;

    int i = left, j = mid + 1;

    while (i <= mid && j <= right) {

        if (arr[i] <= arr[j]) temp.push\_back(arr[i++]);

        else temp.push\_back(arr[j++]);

    }

    while (i <= mid) temp.push\_back(arr[i++]);

    while (j <= right) temp.push\_back(arr[j++]);

    for (int k = 0; k < temp.size(); k++) arr[left + k] = temp[k];

}

void mergeSort(std::vector<int> &arr, int left, int right) {

    if (left >= right) return;

    int mid = left + (right - left) / 2;

    mergeSort(arr, left, mid);

    mergeSort(arr, mid + 1, right);

    merge(arr, left, mid, right);

}

// Đo thời gian chạy Merge Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    mergeSort(data, 0, data.size() - 1);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ test

In put: 9 4 2 5 4 2 5 3

Out put: 2 2 3 4 4 5 5 9

+ Thời gian thực hiện: với dữ liệu nhỏ là 0ms với dữ liệu lớn là 120ms

* **Heap sort (sắp xếp cây)**

+ Ý tưởng: xây dựng 1 cây nhị phân rồi lấy phần tử lớn nhất ra

+ Code

#include <bits/stdc++.h>

#include <fstream>

#include <sstream>

// Đọc dữ liệu từ file văn bản

std::vector<int> readData(const std::string &filename) {

    std::ifstream file(filename);

    std::vector<int> data;

    std::string line;

    if (file.is\_open()) {

        while (std::getline(file, line)) {

            std::stringstream ss(line);

            int num;

            while (ss >> num) {

                data.push\_back(num);

            }

        }

        file.close();

    }

    return data;

}

// Ghi kết quả vào file

void writeData(const std::string &filename, const std::vector<int> &arr, double time) {

    std::ofstream file(filename);

    file << "Time: " << time << " ms\n";

    for (int num : arr) file << num << " ";

}

// Heapify một nút

void heapify(std::vector<int> &arr, int n, int i) {

    int largest = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    if (left < n && arr[left] > arr[largest]) largest = left;

    if (right < n && arr[right] > arr[largest]) largest = right;

    if (largest != i) {

        std::swap(arr[i], arr[largest]);

        heapify(arr, n, largest);

    }

}

// Heap Sort

void heapSort(std::vector<int> &arr) {

    int n = arr.size();

    for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {

        std::swap(arr[0], arr[i]);

        heapify(arr, i, 0);

    }

}

// Đo thời gian chạy Heap Sort

void measureSortTime(std::vector<int> data) {

    auto start = std::chrono::high\_resolution\_clock::now();

    heapSort(data);

    auto end = std::chrono::high\_resolution\_clock::now();

    double time\_taken = std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count();

    writeData("output.txt", data, time\_taken);

}

int main() {

    std::vector<int> data = readData("test.docx");

    if (data.empty()) {

        std::cout << "Error: Input file is empty!\n";

        return 1;

    }

    measureSortTime(data);

    std::cout << "Sorting completed! Check output.txt\n";

    return 0;

}

+ Test

Input: 9 4 2 5 4 2 5 3

Out put: 2 2 3 4 4 5 5 9

+ Thời gian thực hiên: 0ms với dữ liệu nhỏ và 100ms với dữ liệu lớn

**Bài 2/ bt ứng dụng**

1. Định nghĩa cấu trúc dữ liệu để lưu trữ các dữ liệu trong bộ nhớ của máy tính

* Để lưu trữ thông tin về mỗi số hạng của một dãy thức bậc n trong bộ nhớ máy tính, ta sử dụng một cấu trúc dữ liệu (struct) với 2 phần sau:

+ Hệ số (coefficient): là một số thực (kiểu double), biểu thị giá trị của số hạng

+ Bậc (expoment): là một số nguyên (kiểu int), có giá trị từ 0 đến 100, biểu thị luy thừa của biến số.

1. Cài đặt chương trình.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Định nghĩa cấu trúc dữ liệu

struct Term {

double coefficient; // Hệ số

int exponent;       // Bậc

};

// Hàm so sánh để sắp xếp theo bậc tăng dần

bool compareTerms(const Term &a, const Term &b) {

return a.exponent < b.exponent;

}

// Hàm hiển thị danh sách các số hạng

void printTerms(const vector<Term> &terms) {

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

cout << terms[i].coefficient << "x^" << terms[i].exponent;

if (i < terms.size() - 1) cout << " + ";

}

cout << endl;

}

int main() {

// Khai báo danh sách số hạng

vector<Term> terms = {

{3.2, 5}, {1.5, 2}, {4.1, 3}, {2.0, 4}, {5.0, 0}

};

    cout << "Dãy thức ban đầu:" << endl;

    printTerms(terms);

    // Sắp xếp dãy thức theo bậc tăng dần

    sort(terms.begin(), terms.end(), compareTerms);

    cout << "Dãy thức sau khi sắp xếp:" << endl;

    printTerms(terms);

    return 0;

}

* In put

Nhập số lượng số hạng: 5

Nhập hệ số và bậc của từng số hạng:

Số hạng 1 (hệ số, bậc): 3.2 5

Số hạng 2 (hệ số, bậc): 1.5 2

Số hạng 3 (hệ số, bậc): 4.1 3

Số hạng 4 (hệ số, bậc): 2.0 4

Số hạng 5 (hệ số, bậc): 5.0 0

* Out put

Dãy thức ban đầu:

3.2x^5 + 1.5x^2 + 4.1x^3 + 2.0x^4 + 5.0x^0

Dãy thức sau khi sắp xếp:

5.0x^0 + 1.5x^2 + 4.1x^3 + 2.0x^4 + 3.2x^5

**Câu 8/bt ứng dụng**

* ý tưởng

+ Khởi tạo mảng B với tất cả các phần tử bằng o

+ Duyệt qua từng phần tử i từ 1 đến N

+ đếm số lượng phần tử đứng trước i mà lớn hơn i

+ Lưu giá trị đếm được vào B[i]

1. Tìm mảng nghịch thế của một hoán vị.

#include <iostream>

#include <vector>

using namespace std;

vector<int> timMangNghichThe(const vector<int>& A) {

    int N = A.size();

    vector<int> B(N, 0);

    for (int i = 0; i < N; i++) {

        int count = 0;

        for (int j = 0; j < i; j++) {

            if (A[j] > A[i]) {

                count++;

            }

        }

        B[i] = count;

    }

    return B;

}

int main() {

    vector<int> A = {5, 9, 1, 8, 2, 6, 4, 7, 3};

    vector<int> B = timMangNghichThe(A);

    cout << "Mang nghich the: ";

    for (int b : B) {

        cout << b << " ";

    }

    return 0;

}

* Input : A = [5, 9, 1, 8, 2, 6, 4, 7, 3]
* Output: B = [2, 3, 6, 4, 0, 2, 2, 1, 0]

1. Viết hàm tìm hoán vị của một nghịch thế

#include <iostream>

#include <vector>

using namespace std;

vector<int> timHoanViTuNghichThe(const vector<int>& B) {

    int N = B.size();

    vector<int> A(N);

    vector<int> unused; // Các số tự nhiên chưa được sử dụng

    for (int i = 1; i <= N; i++) {

        unused.push\_back(i);

    }

    for (int i = N - 1; i >= 0; i--) {

        A[i] = unused[B[i]];

        unused.erase(unused.begin() + B[i]);

    }

    return A;

}

int main() {

    vector<int> B = {2, 3, 6, 4, 0, 2, 2, 1, 0};

    vector<int> A = timHoanViTuNghichThe(B);

    cout << "Hoan vi tuong ung: ";

    for (int a : A) {

        cout << a << " ";

    }

    return 0;

}

* Input: B = [2, 3, 6, 4, 0, 2, 2, 1, 0]
* Output: A = [5, 9, 1, 8, 2, 6, 4, 7, 3]

**Bài 9\ bt ứng dụng**

a)

#include <iostream>

#include <fstream>

#include <vector>

#include <set>

using namespace std;

struct SinhVien {

    int ma;

    char ho\_dem[21];

    char ten[41];

    int ngay, thang, nam;

    char phai[4];

    float diem;

};

void nhapDanhSach(vector<SinhVien> &danh\_sach) {

    int n;

    set<int> ma\_da\_co;

    cout << "Nhập số lượng sinh viên (tối thiểu 10): ";

    cin >> n;

    cin.ignore();

    while (n < 10) {

        cout << "Phải nhập ít nhất 10 sinh viên! Nhập lại: ";

        cin >> n;

        cin.ignore();

    }

    for (int i = 0; i < n; i++) {

        SinhVien sv;

        do {

            cout << "Nhập mã số sinh viên " << i + 1 << ": ";

            cin >> sv.ma;

            cin.ignore();

            if (ma\_da\_co.count(sv.ma)) {

                cout << "Mã số đã tồn tại! Vui lòng nhập lại.\n";

            }

        } while (ma\_da\_co.count(sv.ma));

        ma\_da\_co.insert(sv.ma);

        cout << "Nhập họ và đệm: ";

        cin.getline(sv.ho\_dem, 21);

        cout << "Nhập tên sinh viên: ";

        cin.getline(sv.ten, 41);

        cout << "Nhập ngày sinh: ";

        cin >> sv.ngay;

        cout << "Nhập tháng sinh: ";

        cin >> sv.thang;

        cout << "Nhập năm sinh: ";

        cin >> sv.nam;

        cin.ignore();

        cout << "Nhập phái (Nam/Nữ): ";

        cin.getline(sv.phai, 4);

        cout << "Nhập điểm trung bình: ";

        cin >> sv.diem;

        cin.ignore();

        danh\_sach.push\_back(sv);

    }

    ofstream file("SINHVIEN.DAT", ios::binary);

    for (const auto &sv : danh\_sach) {

        file.write(reinterpret\_cast<const char\*>(&sv), sizeof(SinhVien));

    }

    file.close();

    cout << "Danh sách đã được lưu vào tệp SINHVIEN.DAT.\n";

}

int main() {

    vector<SinhVien> danh\_sach;

    nhapDanhSach(danh\_sach);

    return 0;

}

b)

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <iomanip>

#include <algorithm>

using namespace std;

struct SinhVien {

    int maSo;

    string hoVaDem;

    string ten;

    int ngaySinh;

    int thangSinh;

    int namSinh;

    string phai;

    float diemTB;

};

void nhapSinhVien(SinhVien &sv, vector<SinhVien> &danhSach) {

    while (true) {

        cout << "Nhập mã số sinh viên: ";

        cin >> sv.maSo;

        bool trungMa = false;

        for (const auto &s : danhSach) {

            if (s.maSo == sv.maSo) {

                trungMa = true;

                break;

            }

        }

        if (!trungMa) break;

        cout << "Mã số sinh viên đã tồn tại, vui lòng nhập lại!\n";

    }

    cin.ignore();

    cout << "Nhập họ và đệm (tối đa 20 ký tự): ";

    getline(cin, sv.hoVaDem);

    if (sv.hoVaDem.length() > 20) sv.hoVaDem = sv.hoVaDem.substr(0, 20);

    cout << "Nhập tên sinh viên (tối đa 40 ký tự): ";

    getline(cin, sv.ten);

    if (sv.ten.length() > 40) sv.ten = sv.ten.substr(0, 40);

    cout << "Nhập ngày sinh: "; cin >> sv.ngaySinh;

    cout << "Nhập tháng sinh: "; cin >> sv.thangSinh;

    cout << "Nhập năm sinh: "; cin >> sv.namSinh;

    do {

        cout << "Nhập phái (Nam/Nữ): ";

        cin >> sv.phai;

    } while (sv.phai != "Nam" && sv.phai != "Nữ");

    do {

        cout << "Nhập điểm trung bình (0.00 - 10.00): ";

        cin >> sv.diemTB;

    } while (sv.diemTB < 0.00 || sv.diemTB > 10.00);

}

void luuSinhVien(const string &tenFile, const vector<SinhVien> &danhSach) {

    ofstream file(tenFile, ios::binary);

    if (!file) {

        cout << "Không thể mở file để ghi dữ liệu!\n";

        return;

    }

    for (const auto &sv : danhSach) {

        file.write(reinterpret\_cast<const char\*>(&sv), sizeof(SinhVien));

    }

    file.close();

    cout << "Dữ liệu sinh viên đã được lưu vào tập tin " << tenFile << "\n";

}

vector<SinhVien> docSinhVien(const string &tenFile) {

    vector<SinhVien> danhSach;

    ifstream file(tenFile, ios::binary);

    if (!file) {

        cout << "Không thể mở file để đọc dữ liệu!\n";

        return danhSach;

    }

    SinhVien sv;

    while (file.read(reinterpret\_cast<char\*>(&sv), sizeof(SinhVien))) {

        danhSach.push\_back(sv);

    }

    file.close();

    return danhSach;

}

void inDanhSach(const vector<SinhVien> &danhSach) {

    cout << "\nDanh sách sinh viên:\n";

    cout << left << setw(10) << "Mã số" << setw(25) << "Họ và đệm" << setw(15) << "Tên"

         << setw(10) << "Ngày sinh" << setw(10) << "Phái" << setw(10) << "Điểm TB" << endl;

    cout << string(80, '-') << endl;

    for (const auto &sv : danhSach) {

        cout << left << setw(10) << sv.maSo << setw(25) << sv.hoVaDem << setw(15) << sv.ten

             << setw(2) << sv.ngaySinh << "/" << setw(2) << sv.thangSinh << "/" << setw(4) << sv.namSinh

             << setw(10) << sv.phai << setw(10) << sv.diemTB << endl;

    }

}

int main() {

    vector<SinhVien> danhSach;

    int soLuong = 10;

    cout << "Nhập danh sách " << soLuong << " sinh viên:\n";

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

        SinhVien sv;

        nhapSinhVien(sv, danhSach);

        danhSach.push\_back(sv);

    }

    luuSinhVien("SINHVIEN.DAT", danhSach);

    // Đọc dữ liệu từ tập tin

    danhSach = docSinhVien("SINHVIEN.DAT");

    // Sắp xếp danh sách theo mã số sinh viên tăng dần

    sort(danhSach.begin(), danhSach.end(), [](const SinhVien &a, const SinhVien &b) {

        return a.maSo < b.maSo;

    });

    // In danh sách đã sắp xếp

    inDanhSach(danhSach);

    return 0;

}