**Bài tập ứng dụng 9 c, d:**

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

using namespace std;

//Cấu trúc sinh viên

struct SinhVien {

    int MaSo;

    string HoVaDem;

    string Ten;

    int NgaySinh;

    int ThangSinh;

    int NamSinh;

    string Phai;

    float DiemTrungBinh;

};

// Ghi thông tin sinh viên vào tập tin

void GhiVaoFile(SinhVien ds[], int n, const string& TenFile) {

    ofstream file(TenFile);

    if (!file) {

        cerr << "Khong the mo file!" << endl;

        return;

    }

    for (int i = 0; i < n; i++) {

        file << ds[i].MaSo << " " << ds[i].HoVaDem << " " << ds[i].Ten << " "

             << ds[i].NgaySinh << " " << ds[i].ThangSinh << " " << ds[i].NamSinh

             << " " << ds[i].Phai << " " << fixed << setprecision(2) << ds[i].DiemTrungBinh << endl;

    }

    file.close();

}

//Hàm Merge cho Merge Sort

void Merge(SinhVien a[], int left, int mid, int right, bool (\*cmp)(const SinhVien&, const SinhVien&)) {

    int n1 = mid - left + 1;

    int n2 = right - mid;

    SinhVien\* leftArr = new SinhVien[n1];

    SinhVien\* rightArr = new SinhVien[n2];

    for (int i = 0; i < n1; i++)

        leftArr[i] = a[left + i];

    for (int i = 0; i < n2; i++)

        rightArr[i] = a[mid + 1 + i];

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {

        if (cmp(leftArr[i], rightArr[j])) {

            a[k] = leftArr[i];

            i++;

        } else {

            a[k] = rightArr[j];

            j++;

        }

        k++;

    }

    while (i < n1) {

        a[k] = leftArr[i];

        i++;

        k++;

    }

    while (j < n2) {

        a[k] = rightArr[j];

        j++;

        k++;

    }

    delete[] leftArr;

    delete[] rightArr;

}

// Sắp xếp theo mã sinh viên

bool SapXepTheoMaSinhVien(const SinhVien& sv1, const SinhVien& sv2) {

    return sv1.MaSo < sv2.MaSo;

}

// Sắp xếp theo tên sinh viên và họ đệm

bool SapXepTheoTenSinhVien(const SinhVien& sv1, const SinhVien& sv2) {

    if (sv1.Ten == sv2.Ten) {

        return sv1.HoVaDem < sv2.HoVaDem;

    }

    return sv1.Ten < sv2.Ten;

}

// Sắp xếp theo điểm trung bình

bool SapXepTheoDiemTrungBinh(const SinhVien& sv1, const SinhVien& sv2) {

    return sv1.DiemTrungBinh > sv2.DiemTrungBinh;

}

// Sử dụng thuật toán Merge Sort

void MergeSort(SinhVien a[], int left, int right, bool (\*cmp)(const SinhVien&, const SinhVien&)) {

    if (left < right) {

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

        MergeSort(a, left, mid, cmp);

        MergeSort(a, mid + 1, right, cmp);

        Merge(a, left, mid, right, cmp);

    }

}

int main() {

    SinhVien DanhSachSinhVien[3] = {

        {101, "Nguyen Van", "A", 1, 1, 2000, "Nam", 9},

        {102, "Le Thi", "B", 2, 2, 2002, "Nu", 8},

        {103, "Tran Van", "C", 3, 5, 2001, "Nam", 7.5}

    };

    int n = 3;

    MergeSort(DanhSachSinhVien, 0, n - 1, SapXepTheoMaSinhVien);

    GhiVaoFile(DanhSachSinhVien, n, "SVMASO.IDX");

    MergeSort(DanhSachSinhVien, 0, n - 1, SapXepTheoTenSinhVien);

    GhiVaoFile(DanhSachSinhVien, n, "SVTH.IDX");

    MergeSort(DanhSachSinhVien, 0, n - 1, SapXepTheoDiemTrungBinh);

    GhiVaoFile(DanhSachSinhVien, n, "SVDTB.IDX");

    cout << "Da tao cac tap tin chi muc: SVMASO.IDX, SVTH.IDX, SVDTB.IDX" << endl;

    return 0;

}

Ý niệm: Nhập thông tin cho một sinh viên bao gồm mã số, họ đệm, tên, ngày, tháng, năm sinh, phái và điểm trung bình sau đó sử dụng thuật toán Merge Sort để chia thông tin sinh viên ra thành các mảng rồi hợp lại và sắp xếp theo yêu cầu.

Vd với bài trên:

Kết quả cho ra:

SVDTB.IDX

101 Nguyen Van A 1 1 2000 Nam 9.00

102 Le Thi B 2 2 2002 Nu 8.00

103 Tran Van C 3 5 2001 Nam 7.50

SVMASO.IDX

101 Nguyen Van A 1 1 2000 Nam 9.00

102 Le Thi B 2 2 2002 Nu 8.00

103 Tran Van C 3 5 2001 Nam 7.50

SVTH.IDX

101 Nguyen Van A 1 1 2000 Nam 9.00

102 Le Thi B 2 2 2002 Nu 8.00

103 Tran Van C 3 5 2001 Nam 7.50