#include <iostream>

#include <cstring>

#include <cctype>

using namespace std;

struct Pegawai {

string NIP;

string Nama;

string Alamat;

char Golongan;

};

const int MAX\_PEGAWAI = 5;

Pegawai dataPegawai[MAX\_PEGAWAI] = {

{"D003", "Kholid", "Surabaya", 'A'},

{"D005", "Safrodin", "Surabaya", 'A'},

{"D001", "Artiani", "Surabaya", 'A'},

{"D004", "Rizky", "Surabaya", 'A'},

{"D002", "Fahim", "Surabaya", 'A'}

};

void swap(Pegawai& a, Pegawai& b) {

Pegawai temp = a;

a = b;

b = temp;

}

void quickSortNIP(Pegawai arr[], int left, int right, bool ascending) {

int i = left, j = right;

string pivot = arr[(left + right) / 2].NIP;

while (i <= j) {

while (ascending ? arr[i].NIP < pivot : arr[i].NIP > pivot)

i++;

while (ascending ? arr[j].NIP > pivot : arr[j].NIP < pivot)

j--;

if (i <= j) {

swap(arr[i], arr[j]);

i++;

j--;

}

}

if (left < j)

quickSortNIP(arr, left, j, ascending);

if (i < right)

quickSortNIP(arr, i, right, ascending);

}

void quickSortNama(Pegawai arr[], int left, int right, bool ascending) {

int i = left, j = right;

string pivot = arr[(left + right) / 2].Nama;

while (i <= j) {

while (ascending ? arr[i].Nama < pivot : arr[i].Nama > pivot)

i++;

while (ascending ? arr[j].Nama > pivot : arr[j].Nama < pivot)

j--;

if (i <= j) {

swap(arr[i], arr[j]);

i++;

j--;

}

}

if (left < j)

quickSortNama(arr, left, j, ascending);

if (i < right)

quickSortNama(arr, i, right, ascending);

}

void merge(Pegawai temp[], int left, int mid, int right, bool ascending, bool byNIP) {

int i = left, j = mid + 1, k = left;

while (i <= mid && j <= right) {

if (byNIP) {

if (ascending ? temp[i].NIP < temp[j].NIP : temp[i].NIP > temp[j].NIP)

dataPegawai[k++] = temp[i++];

else

dataPegawai[k++] = temp[j++];

}

else {

if (ascending ? temp[i].Nama < temp[j].Nama : temp[i].Nama > temp[j].Nama)

dataPegawai[k++] = temp[i++];

else

dataPegawai[k++] = temp[j++];

}

}

while (i <= mid)

dataPegawai[k++] = temp[i++];

while (j <= right)

dataPegawai[k++] = temp[j++];

}

void mergeSortNIP(Pegawai temp[], int left, int right, bool ascending) {

if (left < right) {

int mid = (left + right) / 2;

mergeSortNIP(temp, left, mid, ascending);

mergeSortNIP(temp, mid + 1, right, ascending);

merge(temp, left, mid, right, ascending, true);

}

}

void mergeSortNama(Pegawai temp[], int left, int right, bool ascending) {

if (left < right) {

int mid = (left + right) / 2;

mergeSortNama(temp, left, mid, ascending);

mergeSortNama(temp, mid + 1, right, ascending);

merge(temp, left, mid, right, ascending, false);

}

}

int main() {

int choice, sortBy;

bool ascending;

Pegawai temp[MAX\_PEGAWAI];

cout << "Data Pegawai:" << endl;

for (int i = 0; i < MAX\_PEGAWAI; i++) {

cout << "NIP: " << dataPegawai[i].NIP << ", Nama: " << dataPegawai[i].Nama

<< ", Alamat: " << dataPegawai[i].Alamat << ", Golongan: " << dataPegawai[i].Golongan << endl;

}

cout << "\nPilih metode pengurutan:" << endl;

cout << "1. Quick Sort" << endl;

cout << "2. Merge Sort" << endl;

cout << "Masukkan pilihan (1 atau 2): ";

cin >> choice;

if (choice != 1 && choice != 2) {

cout << "Pilihan tidak valid!" << endl;

return 0;

}

cout << "\nPilih urutan:" << endl;

cout << "1. Naik" << endl;

cout << "2. Turun" << endl;

cout << "Masukkan pilihan (1 atau 2): ";

cin >> choice;

ascending = (choice == 1);

cout << "\nPilih kriteria pengurutan:" << endl;

cout << "1. NIP" << endl;

cout << "2. Nama" << endl;

cout << "Masukkan pilihan (1 atau 2): ";

cin >> sortBy;

for (int i = 0; i < MAX\_PEGAWAI; i++)

temp[i] = dataPegawai[i];

if (sortBy == 1) {

if (choice == 1)

quickSortNIP(dataPegawai, 0, MAX\_PEGAWAI - 1, ascending);

else

mergeSortNIP(temp, 0, MAX\_PEGAWAI - 1, ascending);

}

else if (sortBy == 2) {

if (choice == 1)

quickSortNama(dataPegawai, 0, MAX\_PEGAWAI - 1, ascending);

else

mergeSortNama(temp, 0, MAX\_PEGAWAI - 1, ascending);

}

else {

cout << "Pilihan tidak valid!" << endl;

return 0;

}

cout << "\nData Pegawai setelah diurutkan:" << endl;

for (int i = 0; i < MAX\_PEGAWAI; i++) {

cout << "NIP: " << dataPegawai[i].NIP << ", Nama: " << dataPegawai[i].Nama

<< ", Alamat: " << dataPegawai[i].Alamat << ", Golongan: " << dataPegawai[i].Golongan << endl;

}

return 0;

}

Kesimpulan pada percobaan kali ini adalah **Quick Sort** lebih disukai untuk data yang besar dan membutuhkan kecepatan pengurutan yang tinggi, sedangkan **Merge Sort** lebih cocok untuk data yang membutuhkan stabilitas dan memiliki akses ke komputasi paralel.