#include <iostream>

#include <string>

using namespace std;

const int MAX\_PEGAWAI = 5; // Jumlah maksimum pegawai

// Struktur data untuk menyimpan data pegawai

struct Pegawai {

string NIP;

string Nama;

string Alamat;

char Golongan;

};

// Fungsi untuk menukar data pegawai

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

Pegawai temp = a;

a = b;

b = temp;

}

// Implementasi Straight Insertion Sort berdasarkan NIP (ascending)

void insertionSortNIPAsc(Pegawai data[], int n) {

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

Pegawai temp = data[i];

int j = i - 1;

while (j >= 0 && data[j].NIP > temp.NIP) {

data[j + 1] = data[j];

j--;

}

data[j + 1] = temp;

}

}

// Implementasi Straight Insertion Sort berdasarkan NIP (descending)

void insertionSortNIPDesc(Pegawai data[], int n) {

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

Pegawai temp = data[i];

int j = i - 1;

while (j >= 0 && data[j].NIP < temp.NIP) {

data[j + 1] = data[j];

j--;

}

data[j + 1] = temp;

}

}

// Implementasi Straight Insertion Sort berdasarkan Nama (ascending)

void insertionSortNamaAsc(Pegawai data[], int n) {

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

Pegawai temp = data[i];

int j = i - 1;

while (j >= 0 && data[j].Nama > temp.Nama) {

data[j + 1] = data[j];

j--;

}

data[j + 1] = temp;

}

}

// Implementasi Straight Insertion Sort berdasarkan Nama (descending)

void insertionSortNamaDesc(Pegawai data[], int n) {

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

Pegawai temp = data[i];

int j = i - 1;

while (j >= 0 && data[j].Nama < temp.Nama) {

data[j + 1] = data[j];

j--;

}

data[j + 1] = temp;

}

}

// Implementasi Selection Sort berdasarkan NIP (ascending)

void selectionSortNIPAsc(Pegawai data[], int n) {

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

int minIndex = i;

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

if (data[j].NIP < data[minIndex].NIP) {

minIndex = j;

}

}

tukarPegawai(data[i], data[minIndex]);

}

}

// Implementasi Selection Sort berdasarkan NIP (descending)

void selectionSortNIPDesc(Pegawai data[], int n) {

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

int maxIndex = i;

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

if (data[j].NIP > data[maxIndex].NIP) {

maxIndex = j;

}

}

tukarPegawai(data[i], data[maxIndex]);

}

}

// Implementasi Selection Sort berdasarkan Nama (ascending)

void selectionSortNamaAsc(Pegawai data[], int n) {

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

int minIndex = i;

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

if (data[j].Nama < data[minIndex].Nama) {

minIndex = j;

}

}

tukarPegawai(data[i], data[minIndex]);

}

}

// Implementasi Selection Sort berdasarkan Nama (descending)

void selectionSortNamaDesc(Pegawai data[], int n) {

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

int maxIndex = i;

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

if (data[j].Nama > data[maxIndex].Nama) {

maxIndex = j;

}

}

tukarPegawai(data[i], data[maxIndex]);

}

}

// Fungsi untuk menampilkan data pegawai

void tampilkanData(Pegawai data[], int n) {

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

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

cout << "NIP: " << data[i].NIP << ", Nama: " << data[i].Nama << ", Alamat: " << data[i].Alamat << ", Golongan: " << data[i].Golongan << endl;

}

cout << endl;

}

int main() {

Pegawai dataPegawai[MAX\_PEGAWAI] = {

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

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

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

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

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

};

int pilihan, urutan;

cout << "Pilih metode pengurutan:" << endl;

cout << "1. Insertion Sort berdasarkan NIP" << endl;

cout << "2. Insertion Sort berdasarkan Nama" << endl;

cout << "3. Selection Sort berdasarkan NIP" << endl;

cout << "4. Selection Sort berdasarkan Nama" << endl;

cout << "Masukkan pilihan (1-4): ";

cin >> pilihan;

cout << "Pilih urutan pengurutan:" << endl;

cout << "1. Naik (Ascending)" << endl;

cout << "2. Turun (Descending)" << endl;

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

cin >> urutan;

switch (pilihan) {

case 1:

if (urutan == 1) {

insertionSortNIPAsc(dataPegawai, MAX\_PEGAWAI);

cout << "Data setelah diurutkan dengan Insertion Sort berdasarkan NIP (Ascending):" << endl;

}

else {

insertionSortNIPDesc(dataPegawai, MAX\_PEGAWAI);

cout << "Data setelah diurutkan dengan Insertion Sort berdasarkan NIP (Descending):" << endl;

}

tampilkanData(dataPegawai, MAX\_PEGAWAI);

break;

case 2:

if (urutan == 1) {

insertionSortNamaAsc(dataPegawai, MAX\_PEGAWAI);

cout << "Data setelah diurutkan dengan Insertion Sort berdasarkan Nama (Ascending):" << endl;

}

else {

}

}

}

Kesimpulan pada percobaan ini adalah **Insertion Sort** lebih disukai untuk data yang kecil dan hampir terurut, sedangkan **Selection Sort** lebih cocok untuk data yang kecil dan membutuhkan stabilitas.