Praktikum 7

Bubble sort dan Shell sort

Nama: Dio Stania Adinata

NRP: 5223600024

latihan 1

Input:

#include <iostream>

#include <cstdlib>

#define MAX 10

int Data[MAX];

using namespace std;

// Prosedur menukar data

void Tukar(int\* a, int\* b)

{

    int temp;

    temp = \*a;

    \*a = \*b;

    \*b = temp;

}

// Fungsi Bubble Sort

void BubbleSort()

{

    cout << "\n\nBUBBLE SORT PROCESS:" << endl;

    for (int i = 0; i < MAX - 1; i++)

    {

        bool swapped = false;

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

        {

            if (Data[j] > Data[j + 1])

            {

                Tukar(&Data[j], &Data[j + 1]);

                swapped = true;

            }

        }

        if (!swapped) break; //Jika tidak ada lagi pertukaran maka pengurutan sudah selesai

        cout << "Iteration " << i + 1 << ": ";

        for (int k = 0; k < MAX; k++) {

            cout << Data[k] << " ";

        }

        cout << endl;

    }

}

//Fungsi Shell Sort

void ShellSort()

{

    cout << "\n\nSHELL SORT PROCESS:" << endl;

    int Jarak;

    for (Jarak = MAX / 2; Jarak > 0; Jarak /= 2)

    {

        for (int i = Jarak; i < MAX; i += 1)

        {

            int temp = Data[i];

            int j;

            for (j = i; j >= Jarak && Data[j - Jarak] > temp; j -= Jarak)

            {

                Data[j] = Data[j - Jarak];

            }

            Data[j] = temp;

        }

        cout << "Gap " << Jarak << ": ";

        for (int k = 0; k < MAX; k++) {

            cout << Data[k] << " ";

        }

        cout << endl;

    }

}

int main()

{

    srand(0);

    //Untuk menginisialisasi bilang acak

    cout << "DATA SEBELUM TERURUT" << endl;

    for (int i = 0; i < MAX; i++)

    {

        Data[i] = (int)rand() / 1000 + 1;

        cout << "Data ke " << i << " : " << Data[i] << endl;

    }

    // Memanggil fungsi BubbleSort untuk mengurutkan data

    BubbleSort();

    // Memanggil fungsi ShellSort untuk mengurutkan data

    ShellSort();

    // Menampilkan data setelah diurutkan

    cout << "\nDATA SETELAH TERURUT" << endl;

    for (int i = 0; i < MAX; i++)

    {

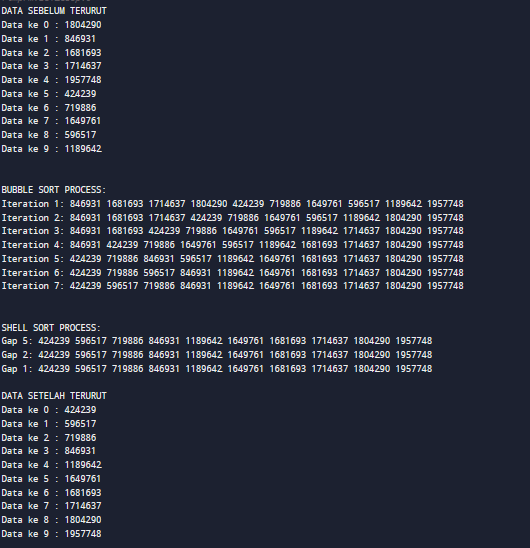
        cout << "Data ke " << i << " : " << Data[i] << endl;

    }

    return 0;

}

Output:



Latihan 2

Input:

#include <iostream>

#include <cstdlib>

#define MAX 10

int Data[MAX];

int comparisonCountBubble = 0;

int shiftCountBubble = 0;

int comparisonCountShell = 0;

int shiftCountShell = 0;

using namespace std;

//fungso untuk menukar data

void Tukar(int\* a, int\* b)

{

    int temp;

    temp = \*a;

    \*a = \*b;

    \*b = temp;

}

//fungsi untuk pengurutan data dengan metode Bubble Sort

void BubbleSort()

{

    cout << "\n\nBUBBLE SORT PROCESS:" << endl;

    for (int i = 0; i < MAX - 1; i++)

    {

        bool swapped = false;

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

        {

            comparisonCountBubble++;

            if (Data[j] > Data[j + 1])

            {

                Tukar(&Data[j], &Data[j + 1]);

                swapped = true;

                shiftCountBubble++;

            }

        }

        if (!swapped) break; // Jika tidak ada lagi pertukaran maka pengurutan sudah selesai

        cout << "Iteration " << i + 1 << ": ";

        for (int k = 0; k < MAX; k++) {

            cout << Data[k] << " ";

        }

        cout << endl;

    }

}

// Prosedur untuk pengurutan data dengan metode Shell Sort

void ShellSort()

{

    cout << "\n\nSHELL SORT PROCESS:" << endl;

    int Jarak;

    for (Jarak = MAX / 2; Jarak > 0; Jarak /= 2)

    {

        for (int i = Jarak; i < MAX; i += 1)

        {

            int temp = Data[i];

            int j;

            for (j = i; j >= Jarak && Data[j - Jarak] > temp; j -= Jarak)

            {

                Data[j] = Data[j - Jarak];

                comparisonCountShell++;

                shiftCountShell++;

            }

            Data[j] = temp;

        }

        cout << "Gap " << Jarak << ": ";

        for (int k = 0; k < MAX; k++) {

            cout << Data[k] << " ";

        }

        cout << endl;

    }

}

int main()

{

    srand(0);

    //Menginisiasi bilangan adcak

    cout << "DATA SEBELUM TERURUT" << endl;

    for (int i = 0; i < MAX; i++)

    {

        Data[i] = (int)rand() / 1000 + 1;

        cout << "Data ke " << i << " : " << Data[i] << endl;

    }

    // Memanggil fungsi BubbleSort untuk mengurutkan data

    BubbleSort();

    // Menampilkan jumlah perbandingan dan pergeseran pada algoritma Bubble Sort

    cout << "\nBUBBLE SORT COMPARISON COUNT: " << comparisonCountBubble << endl;

    cout << "BUBBLE SORT SHIFT COUNT: " << shiftCountBubble << endl;

    // Memanggil fungsi ShellSort untuk mengurutkan data

    ShellSort();

    // Menampilkan jumlah perbandingan dan pergeseran pada algoritma Shell Sort

    cout << "\nSHELL SORT COMPARISON COUNT: " << comparisonCountShell << endl;

    cout << "SHELL SORT SHIFT COUNT: " << shiftCountShell << endl;

    // Menampilkan data setelah diurutkan

    cout << "\nDATA SETELAH TERURUT" << endl;

    for (int i = 0; i < MAX; i++)

    {

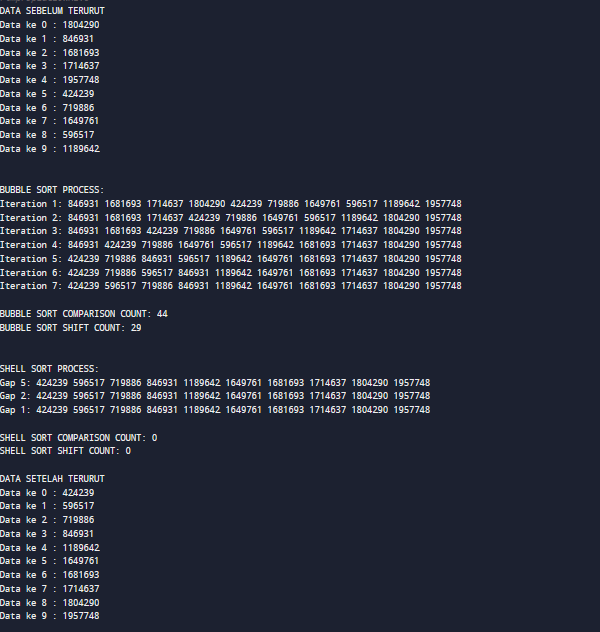
        cout << "Data ke " << i << " : " << Data[i] << endl;

    }

    return 0;

}

output:



Latihan 3

Input:

#include <iostream>

#include <string>

#include <algorithm>

#define MAX 10

using namespace std;

// Struktur data untuk Pegawai

struct Pegawai {

    string NIP;

    string NAMA;

    int UMUR;

    double GAJI;

};

// Fungsi untuk membandingkan Pegawai berdasarkan NIP secara naik

bool compareByNIPAsc(const Pegawai& a, const Pegawai& b) {

    return a.NIP < b.NIP;

}

// Fungsi untuk membandingkan Pegawai berdasarkan NIP secara turun

bool compareByNIPDesc(const Pegawai& a, const Pegawai& b) {

    return a.NIP > b.NIP;

}

// Fungsi untuk membandingkan Pegawai berdasarkan NAMA secara naik

bool compareByNamaAsc(const Pegawai& a, const Pegawai& b) {

    return a.NAMA < b.NAMA;

}

// Fungsi untuk membandingkan Pegawai berdasarkan NAMA secara turun

bool compareByNamaDesc(const Pegawai& a, const Pegawai& b) {

    return a.NAMA > b.NAMA;

}

int main()

{

    Pegawai dataPegawai[MAX] = {

        {"395", "Dio", 19, 5000},

        {"841", "Nauval", 22, 6000},

        {"611", "Rakha", 54, 5500},

        {"159", "Jagur", 26, 5200},

        {"048", "Lauresta", 18, 6500},

        {"753", "Febby", 24, 4800},

        {"479", "Adelia", 29, 5300},

        {"406", "Ryan Gosling", 48, 5800},

        {"124", "Jesse", 31, 5700},

        {"898", "Amanda", 27, 5100}

    };

    int choice;

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

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

    cout << "2. Pengurutan berdasarkan NAMA" << endl;

    cout << "Pilih:" << endl;

    cin >> choice;

    int sortOrder;

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

    cout << "1. Naik" << endl;

    cout << "2. Turun" << endl;

    cout << "Pilih:" << endl;

    cin >> sortOrder;

    // Memilih fungsi pembanding berdasarkan pilihan pengguna

    bool (\*compareFunction)(const Pegawai&, const Pegawai&);

    if (choice == 1) {

        if (sortOrder == 1) {

            compareFunction = compareByNIPAsc;

        }

        else {

            compareFunction = compareByNIPDesc;

        }

    }

    else {

        if (sortOrder == 1) {

            compareFunction = compareByNamaAsc;

        }

        else {

            compareFunction = compareByNamaDesc;

        }

    }

    // Melakukan pengurutan menggunakan fungsi pembanding yang dipilih

    sort(dataPegawai, dataPegawai + MAX, compareFunction);

    // Menampilkan data setelah diurutkan

    cout << "\nDATA PEGAWAI SETELAH DIURUTKAN" << endl;

    for (int i = 0; i < MAX; i++)

    {

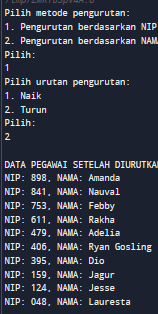
        cout << "NIP: " << dataPegawai[i].NIP << ", NAMA: " << dataPegawai[i].NAMA << endl;

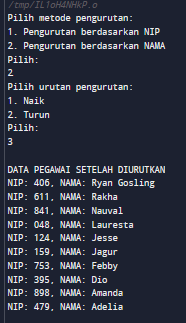
    }

    return 0;

}

Output:





Latihan 4

Bubble Sort dan Shell Sort merupakan dua algoritma pengurutan yang umum digunakan. Bubble Sort bekerja dengan cara membandingkan dan menukar elemen yang berdekatan berulang kali, sedangkan Shell Sort menggunakan konsep gap atau jarak untuk mengurutkan subkelompok data secara terpisah dengan pergeseran yang lebih efisien.

Meskipun Bubble Sort lebih sederhana, Shell Sort umumnya lebih unggul dalam menangani data yang besar karena efisiensi pergeserannya.