Insertion and Selection Sort

Praktikum 06

Nama : Toriq Mardlatillah

NRP : 5223600012

- Percobaan 1

```cpp
#include <iostream>
#include <cstdlib>

#define MAX 10
int Data[MAX];

using namespace std;

// Function for straight insertion sort
void StraightInsertSort()
{
    int i, j, x;
    for (i = 1; i < MAX; i++) {
        x = Data[i];
        j = i - 1;
        while (j >= 0 && x < Data[j]) {
            Data[j + 1] = Data[j];
            j--;
        }
        Data[j + 1] = x;
    }
}
```

```cpp
int main()
{
    int i;
    srand(0);
    // Generating random numbers
    cout << "DATA SEBELUM TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        Data[i] = (int)rand() / 1000 + 1;
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    StraightInsertSort();
    // Sorted data
    cout << "\nDATA SETELAH TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}
```

Output :

```
/tmp/dWG4wV6RXf.o
DATA SEBELUM TERURUT
Data ke 0 : 1804290
Data ke 1 : 846931
Data ke 2 : 1681693
Data ke 3 : 1714637
Data ke 4 : 1957748
Data ke 5 : 424239
Data ke 6 : 719886
Data ke 7 : 1649761
Data ke 8 : 596517
Data ke 9 : 1189642

DATA SETELAH TERURUT
Data ke 0 : 424239
Data ke 1 : 596517
Data ke 2 : 719886
Data ke 3 : 846931
Data ke 4 : 1189642
Data ke 5 : 1649761
Data ke 6 : 1681693
Data ke 7 : 1714637
Data ke 8 : 1804290
Data ke 9 : 1957748


=== Code Execution Successful ===
```

- Percobaan 2 :

```
#include <iostream>
#include <cstdlib>

#define MAX 10
int Data[MAX];

using namespace std;

void BinaryInsertSort()
{
    int i, j, l, r, m, x;
    for (i = 1; i < MAX; i++) {
        x = Data[i];
        l = 0;
        r = i - 1;
        while (l <= r) {
            m = (l + r) / 2;
            if (x < Data[m])
                r = m - 1;
            else
                l = m + 1;
        }
        for (j = i - 1; j >= l; j--)
            Data[j + 1] = Data[j];
        Data[l] = x;
    }
}
```

```
int main()
{
    int i;
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        Data[i] = (int)rand() / 1000 + 1;
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    BinaryInsertSort();
    cout << "\nDATA SETELAH TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}
```

Output :

```
/tmp/uaBLNpSGLY.o
DATA SEBELUM TERURUT
Data ke 0 : 1804290
Data ke 1 : 846931
Data ke 2 : 1681693
Data ke 3 : 1714637
Data ke 4 : 1957748
Data ke 5 : 424239
Data ke 6 : 719886
Data ke 7 : 1649761
Data ke 8 : 596517
Data ke 9 : 1189642

DATA SETELAH TERURUT
Data ke 0 : 424239
Data ke 1 : 596517
Data ke 2 : 719886
Data ke 3 : 846931
Data ke 4 : 1189642
Data ke 5 : 1649761
Data ke 6 : 1681693
Data ke 7 : 1714637
Data ke 8 : 1804290
Data ke 9 : 1957748


=== Code Execution Successful ===
```

- Percobaan 3 :

```
#include <iostream>
#include <cstdlib>

#define MAX 10
int Data[MAX];

using namespace std;


void Tukar(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void SelectionSort()
{
    int i, j, k;
    for (i = 0; i < MAX - 1; i++)
    {
        k = i;
        for (j = i + 1; j < MAX; j++)
            if (Data[k] > Data[j])
                k = j;
        Tukar(&Data[i], &Data[k]);
    }
}
```

```
int main()
{
    int i;
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        Data[i] = (int)rand() / 1000 + 1;
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    SelectionSort();
    cout << "\nDATA SETELAH TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}
```

Output :

```
/tmp/Y1hTLK12un.o
DATA SEBELUM TERURUT
Data ke 0 : 1804290
Data ke 1 : 846931
Data ke 2 : 1681693
Data ke 3 : 1714637
Data ke 4 : 1957748
Data ke 5 : 424239
Data ke 6 : 719886
Data ke 7 : 1649761
Data ke 8 : 596517
Data ke 9 : 1189642

DATA SETELAH TERURUT
Data ke 0 : 424239
Data ke 1 : 596517
Data ke 2 : 719886
Data ke 3 : 846931
Data ke 4 : 1189642
Data ke 5 : 1649761
Data ke 6 : 1681693
Data ke 7 : 1714637
Data ke 8 : 1804290
Data ke 9 : 1957748


=== Code Execution Successful ===
```

- Latihan 1 :

```
#include <iostream>
#include <cstdlib>

// Mendefinisikan ukuran maksimum dari array
#define MAX 10

// Mendeklarasikan array untuk menyimpan data
int Data[MAX];

using namespace std;

// Fungsi untuk menukar dua angka
void Tukar(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

```
// Fungsi untuk melakukan pengurutan dengan penyisipan langsung
void StraightInsertSort()
{
    // Menampilkan header proses
    cout << "\n\nPROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:" << endl;

    // Mengulangi array dimulai dari elemen kedua
    for (int i = 1; i < MAX; i++) {
        int x = Data[i];    // Menyimpan elemen saat ini
        int j = i - 1;      // Memulai perbandingan dengan elemen sebelumnya

        // Memindahkan elemen yang lebih besar dari x ke kanan
        while (j >= 0 && x < Data[j]) {
            Data[j + 1] = Data[j];
            j--;
        }

        // Memasukkan x ke posisi yang tepat
        Data[j + 1] = x;

        // Menampilkan keadaan terkini dari array setelah setiap iterasi
        cout << "Iterasi " << i << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}
```

```
// Fungsi untuk melakukan pengurutan dengan seleksi
void SelectionSort()
{
    // Menampilkan header proses
    cout << "\n\nPROSES PENYORTIRAN DENGAN SELEKSI:" << endl;

    // Mengulangi array
    for (int i = 0; i < MAX - 1; i++)
    {
        int k = i;  // Indeks dari elemen minimum

        // Mencari indeks dari elemen minimum di bagian yang belum diurutkan
        for (int j = i + 1; j < MAX; j++)
            if (Data[k] > Data[j])
                k = j;

        // Menukar elemen saat ini dengan elemen minimum
        Tukar(&Data[i], &Data[k]);

        // Menampilkan keadaan terkini dari array setelah setiap iterasi
        cout << "Iterasi " << i + 1 << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}
```

```
// Fungsi utama
int main()
{
    srand(0);   // Menetapkan seed generator angka acak

    // Menampilkan data awal sebelum diurutkan
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = (int)rand() / 1000 + 1;   // Menghasilkan data acak
        cout << "Data ke " << i << " : " << Data[i] << endl; // Menampilkan data
    }

    // Melakukan pengurutan dengan penyisipan langsung
    StraightInsertSort();

    // Melakukan pengurutan dengan seleksi
    SelectionSort();

    // Menampilkan data yang sudah diurutkan
    cout << "\nDATA SETELAH TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }

    return 0;
}
```

Output :

```
/tmp/ROAKLhVAwS.o
DATA SEBELUM TERURUT
Data ke 0 : 1804290
Data ke 1 : 846931
Data ke 2 : 1681693
Data ke 3 : 1714637
Data ke 4 : 1957748
Data ke 5 : 424239
Data ke 6 : 719886
Data ke 7 : 1649761
Data ke 8 : 596517
Data ke 9 : 1189642


PROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:
Iterasi 1: 846931 1804290 1681693 1714637 1957748 424239 719886 1649761 596517 1189642
Iterasi 2: 846931 1681693 1804290 1714637 1957748 424239 719886 1649761 596517 1189642
Iterasi 3: 846931 1681693 1714637 1804290 1957748 424239 719886 1649761 596517 1189642
Iterasi 4: 846931 1681693 1714637 1804290 1957748 424239 719886 1649761 596517 1189642
Iterasi 5: 424239 846931 1681693 1714637 1804290 1957748 719886 1649761 596517 1189642
Iterasi 6: 424239 719886 846931 1681693 1714637 1804290 1957748 1649761 596517 1189642
Iterasi 7: 424239 719886 846931 1649761 1681693 1714637 1804290 1957748 596517 1189642
Iterasi 8: 424239 596517 719886 846931 1649761 1681693 1714637 1804290 1957748 1189642
Iterasi 9: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
```

```
PROSES PENYORTIRAN DENGAN SELEKSI:
Iterasi 1: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 2: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 3: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 4: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 5: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 6: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 7: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 8: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 9: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748

DATA SETELAH TERURUT
Data ke 0 : 424239
Data ke 1 : 596517
Data ke 2 : 719886
Data ke 3 : 846931
Data ke 4 : 1189642
Data ke 5 : 1649761
Data ke 6 : 1681693
Data ke 7 : 1714637
Data ke 8 : 1804290
Data ke 9 : 1957748
```

- Latihan 2 :

```
#include <iostream>
#include <cstdlib>

#define MAX 10
int Data[MAX];

using namespace std;

int comparisonCount = 0; // Menghitung jumlah perbandingan
int shiftCount = 0;      // Menghitung jumlah pergeseran

void Tukar(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

```
void StraightInsertSort()
{
    cout << "\n\nPROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:" << endl;
    for (int i = 1; i < MAX; i++) {
        int x = Data[i];
        int j = i - 1;
        while (j >= 0 && x < Data[j]) {
            Data[j + 1] = Data[j];
            j--;
            comparisonCount++; // Setiap kali ada perbandingan, tambahkan hitungannya
            shiftCount++;      // Jumlah pergeseran juga bertambah
        }
        Data[j + 1] = x;
        shiftCount++; // Pergeseran tambahan untuk memasukkan x ke posisi yang tepat

        cout << "Iterasi " << i << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
    cout << "Jumlah perbandingan: " << comparisonCount << endl;
    cout << "Jumlah pergeseran: " << shiftCount << endl;
}
```

```
void SelectionSort()
{
    cout << "\n\nPROSES PENYORTIRAN DENGAN SELEKSI:" << endl;
    comparisonCount = 0; // Reset jumlah perbandingan
    shiftCount = 0;      // Reset jumlah pergeseran
    for (int i = 0; i < MAX - 1; i++)
    {
        int k = i;
        for (int j = i + 1; j < MAX; j++) {
            comparisonCount++; // Setiap perbandingan, tambahkan hitungannya
            if (Data[k] > Data[j])
                k = j;
        }
        Tukar(&Data[i], &Data[k]);
        shiftCount++; // Setiap pertukaran, tambahkan hitungannya

        cout << "Iterasi " << i + 1 << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
    cout << "Jumlah perbandingan: " << comparisonCount << endl;
    cout << "Jumlah pergeseran: " << shiftCount << endl;
}
```

```
int main()
{
    srand(0);
    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;
    }

    StraightInsertSort();

    SelectionSort();

    cout << "\nDATA SETELAH TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}
```

Output :

```
DATA SEBELUM TERURUT
Data ke 0 : 1804290
Data ke 1 : 846931
Data ke 2 : 1681693
Data ke 3 : 1714637
Data ke 4 : 1957748
Data ke 5 : 424239
Data ke 6 : 719886
Data ke 7 : 1649761
Data ke 8 : 596517
Data ke 9 : 1189642


PROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:
Iterasi 1: 846931 1804290 1681693 1714637 1957748 424239 719886 1649761 596517 1189642
Iterasi 2: 846931 1681693 1804290 1714637 1957748 424239 719886 1649761 596517 1189642
Iterasi 3: 846931 1681693 1714637 1804290 1957748 424239 719886 1649761 596517 1189642
Iterasi 4: 846931 1681693 1714637 1804290 1957748 424239 719886 1649761 596517 1189642
Iterasi 5: 424239 846931 1681693 1714637 1804290 1957748 719886 1649761 596517 1189642
Iterasi 6: 424239 719886 846931 1681693 1714637 1804290 1957748 1649761 596517 1189642
Iterasi 7: 424239 719886 846931 1649761 1681693 1714637 1804290 1957748 596517 1189642
Iterasi 8: 424239 596517 719886 846931 1649761 1681693 1714637 1804290 1957748 1189642
Iterasi 9: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Jumlah perbandingan: 29
Jumlah pergeseran: 38
```

```
PROSES PENYORTIRAN DENGAN SELEKSI:
Iterasi 1: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 2: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 3: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 4: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 5: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 6: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 7: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 8: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iterasi 9: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Jumlah perbandingan: 45
Jumlah pergeseran: 9

DATA SETELAH TERURUT
Data ke 0 : 424239
Data ke 1 : 596517
Data ke 2 : 719886
Data ke 3 : 846931
Data ke 4 : 1189642
Data ke 5 : 1649761
Data ke 6 : 1681693
Data ke 7 : 1714637
Data ke 8 : 1804290
Data ke 9 : 1957748
```

- Latihan 3 :

```
#include <iostream>
#include <string>
using namespace std;

// Struktur data untuk Pegawai
struct Pegawai {
    string NIP;
    string NAMA;
};

// Metode pengurutan yang dapat dipilih
enum MetodePengurutan {
    BUBBLE_SORT,
    SELECTION_SORT,
    INSERTION_SORT
};

// Pengurutan dapat dipilih secara urut naik atau turun
enum Urutan {
    NAIK,
    TURUN
};

// Pengurutan dapat dipilih berdasarkan NIP dan NAMA
enum Berdasarkan {
    NIP,
    NAMA
};
```

```
// Fungsi untuk tukar dua elemen Pegawai
void Tukar(Pegawai& a, Pegawai& b) {
    Pegawai temp = a;
    a = b;
    b = temp;
}

// Implementasi pengurutan data Pegawai dengan Bubble Sort
void BubbleSort(Pegawai arr[], int n, Urutan urutan, Berdasarkan berdasarkan) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if ((berdasarkan == NIP && arr[j].NIP > arr[j + 1].NIP) ||
                (berdasarkan == NAMA && arr[j].NAMA > arr[j + 1].NAMA)) {
                if (urutan == NAIK) {
                    Tukar(arr[j], arr[j + 1]);
                }
            }
            else {
                if (urutan == TURUN) {
                    Tukar(arr[j], arr[j + 1]);
                }
            }
        }
    }
}
```

```
// Implementasi pengurutan data Pegawai dengan Selection Sort
void SelectionSort(Pegawai arr[], int n, Urutan urutan, Berdasarkan berdasarkan) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if ((berdasarkan == NIP && arr[j].NIP < arr[minIndex].NIP) ||
                (berdasarkan == NAMA && arr[j].NAMA < arr[minIndex].NAMA)) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            if (urutan == NAIK) {
                Tukar(arr[i], arr[minIndex]);
            }
        }
        else {
            if (urutan == TURUN) {
                Tukar(arr[i], arr[minIndex]);
            }
        }
    }
}
```

```
// Implementasi pengurutan data Pegawai dengan Insertion Sort
void InsertionSort(Pegawai arr[], int n, Urutan urutan, Berdasarkan berdasarkan) {
    for (int i = 1; i < n; i++) {
        Pegawai key = arr[i];
        int j = i - 1;
        while (j >= 0 && ((berdasarkan == NIP && arr[j].NIP > key.NIP) ||
            (berdasarkan == NAMA && arr[j].NAMA > key.NAMA))) {
            if (urutan == NAIK) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            else {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
        }
        arr[j + 1] = key;
    }
}
```

```
// Fungsi untuk menampilkan data Pegawai
void TampilkanData(Pegawai arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << "Pegawai ke " << i + 1 << ": NIP = " << arr[i].NIP << ", NAMA = " << arr[i].NAMA
            << endl;
    }
}

int main() {
    const int JUMLAH_PEGAWAI = 5;
    Pegawai dataPegawai[JUMLAH_PEGAWAI] = {
        {"123", "John"},
        {"456", "Alice"},
        {"789", "Bob"},
        {"234", "David"},
        {"567", "Eva"}
    };

    // Pilihan pengguna
    MetodePengurutan metode = BUBBLE_SORT;
    Urutan urutan = NAIK;
    Berdasarkan berdasarkan = NIP;

    cout << "Data Sebelum Diurutkan:" << endl;
    TampilkanData(dataPegawai, JUMLAH_PEGAWAI);
```

```
    // Memilih metode pengurutan
    switch (metode) {
    case BUBBLE_SORT:
        BubbleSort(dataPegawai, JUMLAH_PEGAWAI, urutan, berdasarkan);
        cout << "\nData Setelah Diurutkan dengan Bubble Sort:" << endl;
        break;
    case SELECTION_SORT:
        SelectionSort(dataPegawai, JUMLAH_PEGAWAI, urutan, berdasarkan);
        cout << "\nData Setelah Diurutkan dengan Selection Sort:" << endl;
        break;
    case INSERTION_SORT:
        InsertionSort(dataPegawai, JUMLAH_PEGAWAI, urutan, berdasarkan);
        cout << "\nData Setelah Diurutkan dengan Insertion Sort:" << endl;
        break;
    }

    // Menampilkan data setelah diurutkan
    TampilkanData(dataPegawai, JUMLAH_PEGAWAI);

    return 0;
}
```

Output :

```
/tmp/ADyXghGOKH.o
Data Sebelum Diurutkan:
Pegawai ke 1: NIP = 123, NAMA = John
Pegawai ke 2: NIP = 456, NAMA = Alice
Pegawai ke 3: NIP = 789, NAMA = Bob
Pegawai ke 4: NIP = 234, NAMA = David
Pegawai ke 5: NIP = 567, NAMA = Eva

Data Setelah Diurutkan dengan Bubble Sort:
Pegawai ke 1: NIP = 123, NAMA = John
Pegawai ke 2: NIP = 234, NAMA = David
Pegawai ke 3: NIP = 456, NAMA = Alice
Pegawai ke 4: NIP = 567, NAMA = Eva
Pegawai ke 5: NIP = 789, NAMA = Bob
```

- Latihan 4 :

Insertion dan Selection Sort adalah algoritma pengurutan sederhana dalam mengurutkan elemen yang ada dalam suatu array. Keduanya memiliki sifat dan karakteristik yang perlu dipertimbangkan untuk kecepatan dalam memilah data yang diinginkan.