**Insertion and Selection Sort**

**Nama: Oscar Javier Abdullah**

**Kelas: Gt-A11**

**Percobaan 1**

Input:

```
#include <iostream>
#include <cstdlib>
#define MAX 10

int Data[MAX];

// Fungsi pengurutan penyisipan langsung
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;
    }
}

int main() {
    int i;
    srand(0);

    // Memunculkan bilangan acak
    std::cout << "DATA SEBELUM TERURUT" << std::endl;
    for (i = 0; i < MAX; i++) {
```

```
        Data[i] = rand() % 1000 + 1;
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    StraightInsertSort();

    // Memunculkan Data setelah terurut
    std::cout << "\nDATA SETELAH TERURUT" << std::endl;
    for (i = 0; i < MAX; i++) {
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    return 0;
}
```

Output :

```
/tmp/ywnHm0ZkWT.o
DATA SEBELUM TERURUT
Data ke 0 : 384
Data ke 1 : 887
Data ke 2 : 778
Data ke 3 : 916
Data ke 4 : 794
Data ke 5 : 336
Data ke 6 : 387
Data ke 7 : 493
Data ke 8 : 650
Data ke 9 : 422

DATA SETELAH TERURUT
Data ke 0 : 336
Data ke 1 : 384
Data ke 2 : 387
Data ke 3 : 422
Data ke 4 : 493
Data ke 5 : 650
Data ke 6 : 778
Data ke 7 : 794
Data ke 8 : 887
Data ke 9 : 916
```

**Percobaan 2**

Input:

```
#include <iostream>
#include <cstdlib>
#define MAX 10

int Data[MAX];

// Fungsi pengurutan penyisipan biner
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);
```

```
    // memunculkan bilangan acak
    std::cout << "DATA SEBELUM TERURUT" << std::endl;
    for (i = 0; i < MAX; i++) {
        Data[i] = rand() % 1000 + 1;
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    BinaryInsertSort();

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

Output:

```
/tmp/gcJVx4F5y7.o
DATA SEBELUM TERURUT
Data ke 0 : 384
Data ke 1 : 887
Data ke 2 : 778
Data ke 3 : 916
Data ke 4 : 794
Data ke 5 : 336
Data ke 6 : 387
Data ke 7 : 493
Data ke 8 : 650
Data ke 9 : 422

DATA SETELAH TERURUT
Data ke 0 : 336
Data ke 1 : 384
Data ke 2 : 387
Data ke 3 : 422
Data ke 4 : 493
Data ke 5 : 650
Data ke 6 : 778
Data ke 7 : 794
Data ke 8 : 887
Data ke 9 : 916
```

**Percobaan 3**

Input:

```
#include <iostream>
#include <cstdlib>
#define MAX 10

int Data[MAX];

// Fungsi pertukaran bilangan
void Tukar(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

// Fungsi pengurutan seleksi
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);

    // Membangkitkan bilangan acak
    std::cout << "DATA SEBELUM TERURUT" << std::endl;
    for (i = 0; i < MAX; i++) {
        Data[i] = rand() % 1000 + 1;
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    SelectionSort();

    // Data setelah terurut
    std::cout << "\nDATA SETELAH TERURUT" << std::endl;
    for (i = 0; i < MAX; i++) {
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    return 0;
}
```

Output:

```
/tmp/gcJVx4F5y7.o
DATA SEBELUM TERURUT
Data ke 0 : 384
Data ke 1 : 887
Data ke 2 : 778
Data ke 3 : 916
Data ke 4 : 794
Data ke 5 : 336
Data ke 6 : 387
Data ke 7 : 493
Data ke 8 : 650
Data ke 9 : 422

DATA SETELAH TERURUT
Data ke 0 : 336
Data ke 1 : 384
Data ke 2 : 387
Data ke 3 : 422
Data ke 4 : 493
Data ke 5 : 650
Data ke 6 : 778
Data ke 7 : 794
Data ke 8 : 887
Data ke 9 : 916
```

**Latihan 1**

**1) Tambahkan kode program untuk menampilkan perubahan setiap iterasi dari proses pengurutan dengan penyisipan langsung, penyisipan biner dan seleksi.**

Input:

```
#include <iostream>
#include <cstdlib>
#define MAX 10

int Data[MAX];

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

void TampilkanData() {
    for (int i = 0; i < MAX; i++) {
        std::cout << Data[i] << " ";
    }
    std::cout << std::endl;
}

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;
        TampilkanData();
    }
}

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;
        TampilkanData();
    }
}

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]);
        TampilkanData();
    }
}

int main() {
    srand(0);

    // Membangkitkan bilangan acak
    std::cout << "DATA SEBELUM TERURUT" << std::endl;
    for (int i = 0; i < MAX; i++) {
        Data[i] = rand() % 1000 + 1;
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    std::cout << "\nPengurutan dengan Penyisipan Langsung:" <<
std::endl;
    StraightInsertSort();

    std::cout << "\nPengurutan dengan Penyisipan Biner:" <<
std::endl;
    BinaryInsertSort();

    std::cout << "\nPengurutan dengan Seleksi:" << std::endl;
    SelectionSort();

    return 0;
}
```

Output:

```
/tmp/LFx02zNntp.o
DATA SEBELUM TERURUT
Data ke 0 : 384
Data ke 1 : 887
Data ke 2 : 778
Data ke 3 : 916
Data ke 4 : 794
Data ke 5 : 336
Data ke 6 : 387
Data ke 7 : 493
Data ke 8 : 650
Data ke 9 : 422

Pengurutan dengan Penyisipan Langsung:
384 887 778 916 794 336 387 493 650 422
384 778 887 916 794 336 387 493 650 422
384 778 887 916 794 336 387 493 650 422
384 778 794 887 916 336 387 493 650 422
336 384 778 794 887 916 387 493 650 422
336 384 387 778 794 887 916 493 650 422
336 384 387 493 778 794 887 916 650 422
336 384 387 493 650 778 794 887 916 422
336 384 387 422 493 650 778 794 887 916

Pengurutan dengan Penyisipan Biner:
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916

Pengurutan dengan Seleksi:
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
```

**Latihan 2**

**2) Tambahkan kode program untuk menghitung banyaknya perbandingan dan pergeseran pada algoritma pengurutan penyisipan langsung, penyisipan biner dan seleksi.**

Input:

```
#include <iostream>

#include <cstdlib>

#define MAX 10
```

```
int Data[MAX];
int perbandingan = 0;
int pergeseran = 0;

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

void TampilkanData() {
    for (int i = 0; i < MAX; i++) {
        std::cout << Data[i] << " ";
    }
    std::cout << std::endl;
}

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--;
            perbandingan++;
            pergeseran++;
        }
        Data[j + 1] = x;
        pergeseran++;
        TampilkanData();
    }
```

```
}

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;
            perbandingan++;
        }
        for (j = i - 1; j >= l; j--) {
            Data[j + 1] = Data[j];
            pergeseran++;
        }
        Data[l] = x;
        pergeseran++;
        TampilkanData();
    }
}

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;
            }
            perbandingan++;
        }
        Tukar(&Data[i], &Data[k]);
        pergeseran++;
        TampilkanData();
    }
}

int main() {
    srand(0);

    // Membangkitkan bilangan acak
    std::cout << "DATA SEBELUM TERURUT" << std::endl;
    for (int i = 0; i < MAX; i++) {
        Data[i] = rand() % 1000 + 1;
        std::cout << "Data ke " << i << " : " << Data[i] <<
std::endl;
    }

    std::cout << "\nPengurutan dengan Penyisipan Langsung:" <<
std::endl;
    StraightInsertSort();
    std::cout << "Perbandingan: " << perbandingan << ", Pergeseran:
" << pergeseran << std::endl;

    std::cout << "\nPengurutan dengan Penyisipan Biner:" <<
std::endl;
    BinaryInsertSort();
    std::cout << "Perbandingan: " << perbandingan << ", Pergeseran:
" << pergeseran << std::endl;

    std::cout << "\nPengurutan dengan Seleksi:" << std::endl;
```

```
    SelectionSort();
    std::cout << "Perbandingan: " << perbandingan << ", Pergeseran: " << pergeseran << std::endl;

    return 0;
}
```

Output:

```
/tmp/UvuRoiebpr.o
DATA SEBELUM TERURUT
Data ke 0 : 384
Data ke 1 : 887
Data ke 2 : 778
Data ke 3 : 916
Data ke 4 : 794
Data ke 5 : 336
Data ke 6 : 387
Data ke 7 : 493
Data ke 8 : 650
Data ke 9 : 422

Pengurutan dengan Penyisipan Langsung:
384 887 778 916 794 336 387 493 650 422
384 778 887 916 794 336 387 493 650 422
384 778 887 916 794 336 387 493 650 422
384 778 794 887 916 336 387 493 650 422
336 384 778 794 887 916 387 493 650 422
336 384 387 778 794 887 916 493 650 422
336 384 387 493 778 794 887 916 650 422
336 384 387 493 650 778 794 887 916 422
336 384 387 422 493 650 778 794 887 916
Perbandingan: 26, Pergeseran: 35

Pengurutan dengan Penyisipan Biner:
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
Perbandingan: 51, Pergeseran: 44

Pengurutan dengan Seleksi:
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
336 384 387 422 493 650 778 794 887 916
Perbandingan: 96, Pergeseran: 53
```

**Latihan 3**

**3) Buatlah project baru untuk Latihan dan implementasikan pengurutan data Pegawai**

**pada tugas pendahuluan dengan ketentuan :**

**a. Metode pengurutan dapat dipilih.**

**b. Pengurutan dapat dipilih secara urut naik atau turun.**

**c. Pengurutan dapat dipilih berdasarkan NIP dan NAMA.**

**d. Gunakan struktur data array.**

Input:

```
#include <iostream>
#include <string>
#include <algorithm>

struct Pegawai {
    int NIP;
    std::string NAMA;
};

bool CompareByNIP(const Pegawai& a, const Pegawai& b) {
    return a.NIP < b.NIP;
}

bool CompareByNama(const Pegawai& a, const Pegawai& b) {
    return a.NAMA < b.NAMA;
}

void TampilkanData(const Pegawai* data, int jumlah) {
    for (int i = 0; i < jumlah; i++) {
        std::cout << "NIP: " << data[i].NIP << ", Nama: " << 
data[i].NAMA << std::endl;
    }
}
```

```
int main() {
    const int JUMLAH_PEGAWAI = 5;
    Pegawai dataPegawai[JUMLAH_PEGAWAI] = {
        {123, "Rangga"},
        {456, "Marlon"},
        {789, "Ruben"},
        {234, "Andhika"},
        {567, "Jegel"}
    };

    int pilihanMetode;
    std::cout << "Pilih metode pengurutan:" << std::endl;
    std::cout << "1. Urut berdasarkan NIP" << std::endl;
    std::cout << "2. Urut berdasarkan Nama" << std::endl;
    std::cin >> pilihanMetode;

    int pilihanUrutan;
    std::cout << "Pilih urutan:" << std::endl;
    std::cout << "1. Urut naik" << std::endl;
    std::cout << "2. Urut turun" << std::endl;
    std::cin >> pilihanUrutan;

    switch (pilihanMetode) {
        case 1:
            if (pilihanUrutan == 1) {
                std::sort(dataPegawai, dataPegawai + JUMLAH_PEGAWAI,
CompareByNIP);
            } else {
                std::sort(dataPegawai, dataPegawai + JUMLAH_PEGAWAI,
CompareByNIP);
                std::reverse(dataPegawai, dataPegawai +
JUMLAH_PEGAWAI);
            }
```

```
            break;
        case 2:
            if (pilihanUrutan == 1) {
                std::sort(dataPegawai, dataPegawai + JUMLAH_PEGAWAI,
CompareByNama);
            } else {
                std::sort(dataPegawai, dataPegawai + JUMLAH_PEGAWAI,
CompareByNama);
                std::reverse(dataPegawai, dataPegawai +
JUMLAH_PEGAWAI);
            }
            break;
        default:
            std::cout << "Pilihan metode tidak valid." << std::endl;
            return 1;
    }

    std::cout << "\nData Pegawai setelah diurutkan:" << std::endl;
    TampilkanData(dataPegawai, JUMLAH_PEGAWAI);

    return 0;
}
```

Output:

```
Pilih metode pengurutan:
1. Urut berdasarkan NIP
2. Urut berdasarkan Nama
1
Pilih urutan:
1. Urut naik
2. Urut turun
1

Data Pegawai setelah diurutkan:
NIP: 123, Nama: Rangga
NIP: 234, Nama: Andhika
NIP: 456, Nama: Marlon
NIP: 567, Nama: Jegel
NIP: 789, Nama: Ruben
```

```
/tmp/Y7VD9d3me3.o
Pilih metode pengurutan:
1. Urut berdasarkan NIP
2. Urut berdasarkan Nama
2
Pilih urutan:
1. Urut naik
2. Urut turun
2

Data Pegawai setelah diurutkan:
NIP: 789, Nama: Ruben
NIP: 123, Nama: Rangga
NIP: 456, Nama: Marlon
NIP: 567, Nama: Jegel
NIP: 234, Nama: Andhika
```

**Latihan 4**

**4) Berikan kesimpulan dari percobaan dan latihan yang telah Anda lakukan.**

Memahami Konsep Algoritma Pengurutan:

Dengan praktikum ini, kita dapat memahami bagaimana algoritma pengurutan bekerja, termasuk logika di balik penyisipan langsung, penyisipan biner, dan seleksi.