Praktikum 07

Bubble Sort, Shell Sort

Nama : Tora Sandh Kamulian

NRP : 5223600013

Percobaan 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;
}

// Prosedur pengurutan metode gelembung
void BubbleSort()
{
    for (int i = 0; i < MAX - 1; i++) {
        for (int j = MAX - 1; j >= i; j--) {
            // Membandingkan dua elemen berturut-turut dan menukar posisi jika
perlu
            if (Data[j - 1] > Data[j]) {
                Tukar(&Data[j - 1], &Data[j]);
            }
        }
    }
}

int main()
{
    srand(0);

    // Membangkitkan bilangan 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();

    // 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 :

```
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 :

Input :

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

#define MAX 10
```

```cpp
int Data[MAX];

using namespace std;

// Prosedur menukar data
void Tukar(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

// Prosedur pengurutan metode Shell
void ShellSort()
{
    int Jarak, i, j;
    bool Sudah;
    Jarak = MAX;
    while (Jarak > 1)
    {
        Jarak = Jarak / 2;
        Sudah = false;
        while (!Sudah)
        {
            Sudah = true;
            for (j = 0; j < MAX - Jarak; j++)
            {
                i = j + Jarak;
                if (Data[j] > Data[i])
                {
                    Tukar(&Data[j], &Data[i]);
                    Sudah = false;
                }
            }
        }
    }
}

int main()
{
    srand(0);

    // Membangkitkan bilangan 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 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 :

```
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 :

1. Tambahkan kode program untuk menampilkan perubahan setiap iterasi dari proses pengurutan dengan metode gelembung dan shell.

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;
}

// Prosedur pengurutan 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++)
        {
            if (Data[j] > Data[j + 1])
            {
                Tukar(&Data[j], &Data[j + 1]);
                swapped = true;
            }
        }
        if (!swapped) break; // Jika tidak ada pertukaran, pengurutan sudah
selesai
        cout << "Iteration " << i + 1 << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}

// Prosedur pengurutan 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];
            }
            Data[j] = temp;
        }
        cout << "Gap " << Jarak << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}

int main()
{
    srand(0);

    // Membangkitkan bilangan 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 :

```
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


BUBBLE SORT PROCESS:
Iteration 1: 846931 1681693 1714637 1804290 424239 719886 1649761 596517 1189642 1957748
Iteration 2: 846931 1681693 1714637 424239 719886 1649761 596517 1189642 1804290 1957748
Iteration 3: 846931 1681693 424239 719886 1649761 596517 1189642 1714637 1804290 1957748
Iteration 4: 846931 424239 719886 1649761 596517 1189642 1681693 1714637 1804290 1957748
Iteration 5: 424239 719886 846931 596517 1189642 1649761 1681693 1714637 1804290 1957748
Iteration 6: 424239 719886 596517 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iteration 7: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748


SHELL SORT PROCESS:
Gap 5: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Gap 2: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Gap 1: 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


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

Latihan 2 :

2. Tambahkan kode program untuk menghitung banyaknya perbandingan dan pergeseran pada algoritma gelembung dan shell.

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;

// Prosedur menukar data
void Tukar(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

// Prosedur pengurutan 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 pertukaran, pengurutan sudah
selesai
        cout << "Iteration " << i + 1 << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}

// Prosedur pengurutan 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);

    // Membangkitkan bilangan 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();

    // 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 :

```
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


BUBBLE SORT PROCESS:
Iteration 1: 846931 1681693 1714637 1804290 424239 719886 1649761 596517 1189642 1957748
Iteration 2: 846931 1681693 1714637 424239 719886 1649761 596517 1189642 1804290 1957748
Iteration 3: 846931 1681693 424239 719886 1649761 596517 1189642 1714637 1804290 1957748
Iteration 4: 846931 424239 719886 1649761 596517 1189642 1681693 1714637 1804290 1957748
Iteration 5: 424239 719886 846931 596517 1189642 1649761 1681693 1714637 1804290 1957748
Iteration 6: 424239 719886 596517 846931 1189642 1649761 1681693 1714637 1804290 1957748
Iteration 7: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748

BUBBLE SORT COMPARISON COUNT: 44
BUBBLE SORT SHIFT COUNT: 29


SHELL SORT PROCESS:
Gap 5: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Gap 2: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748
Gap 1: 424239 596517 719886 846931 1189642 1649761 1681693 1714637 1804290 1957748

SHELL SORT COMPARISON COUNT: 0
SHELL SORT SHIFT COUNT: 0

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 :

3. Tambahkan pada project Latihan pada praktikum 7 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>

#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] = {
        {"123", "John Doe", 30, 5000},
        {"456", "Jane Smith", 35, 6000},
        {"789", "David Brown", 40, 5500},
        {"234", "Sarah Johnson", 25, 5200},
        {"567", "Michael Williams", 45, 6500},
        {"890", "Emily Jones", 28, 4800},
        {"345", "Christopher Lee", 32, 5300},
        {"678", "Jessica Davis", 38, 5800},
        {"901", "Daniel Martinez", 33, 5700},
        {"012", "Amanda Wilson", 29, 5100}
    };

    int choice;
    cout << "Pilih metode pengurutan:" << endl;
    cout << "1. Pengurutan berdasarkan NIP" << endl;
    cout << "2. Pengurutan berdasarkan NAMA" << endl;
    cin >> choice;
```

```
    int sortOrder;
    cout << "Pilih urutan pengurutan:" << endl;
    cout << "1. Naik" << endl;
    cout << "2. Turun" << 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 :

a.

```
  int choice;
  cout << "Pilih metode pengurutan:" << endl;
  cout << "1. Pengurutan berdasarkan NIP" << endl;
  cout << "2. Pengurutan berdasarkan NAMA" << endl;
  cin >> choice;
```

```
/tmp/gn5SapP6ol.o
Pilih metode pengurutan:
1. Pengurutan berdasarkan NIP
2. Pengurutan berdasarkan NAMA
1
```

b.

```
int sortOrder;
cout << "Pilih urutan pengurutan:" << endl;
cout << "1. Naik" << endl;
cout << "2. Turun" << endl;
cin >> sortOrder;
```

```
Pilih urutan pengurutan:
1. Naik
2. Turun
1
```

c. Pengaturan dapat diurut berdasarkan NIP atau Nama :

- NIP

```
Pilih metode pengurutan:
1. Pengurutan berdasarkan NIP
2. Pengurutan berdasarkan NAMA
1
Pilih urutan pengurutan:
1. Naik
2. Turun
1

DATA PEGAWAI SETELAH DIURUTKAN
NIP: 012, NAMA: Amanda Wilson
NIP: 123, NAMA: John Doe
NIP: 234, NAMA: Sarah Johnson
NIP: 345, NAMA: Christopher Lee
NIP: 456, NAMA: Jane Smith
NIP: 567, NAMA: Michael Williams
NIP: 678, NAMA: Jessica Davis
NIP: 789, NAMA: David Brown
NIP: 890, NAMA: Emily Jones
NIP: 901, NAMA: Daniel Martinez


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

- Nama

```
Pilih metode pengurutan:
1. Pengurutan berdasarkan NIP
2. Pengurutan berdasarkan NAMA
2
Pilih urutan pengurutan:
1. Naik
2. Turun
2

DATA PEGAWAI SETELAH DIURUTKAN
NIP: 234, NAMA: Sarah Johnson
NIP: 567, NAMA: Michael Williams
NIP: 123, NAMA: John Doe
NIP: 678, NAMA: Jessica Davis
NIP: 456, NAMA: Jane Smith
NIP: 890, NAMA: Emily Jones
NIP: 789, NAMA: David Brown
NIP: 901, NAMA: Daniel Martinez
NIP: 345, NAMA: Christopher Lee
NIP: 012, NAMA: Amanda Wilson


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

d. Gunakan Struktur data Array :

```
Pegawai dataPegawai[MAX] = {
    {"123", "John Doe", 30, 5000},
    {"456", "Jane Smith", 35, 6000},
    {"789", "David Brown", 40, 5500},
    {"234", "Sarah Johnson", 25, 5200},
    {"567", "Michael Williams", 45, 6500},
    {"890", "Emily Jones", 28, 4800},
    {"345", "Christopher Lee", 32, 5300},
    {"678", "Jessica Davis", 38, 5800},
    {"901", "Daniel Martinez", 33, 5700},
    {"012", "Amanda Wilson", 29, 5100}
};
```

4. Kesimpulan :

Bubble Sort adalah algoritma pengurutan sederhana yang membandingkan dan menukar elemen-elemen berdekatan secara berulang, sementara Shell Sort menggunakan konsep gap atau jarak untuk mengurutkan subkelompok data secara terpisah, dengan pergeseran yang lebih efisien.