Bubble, Shell Sort

Nama: Oscar Javier Abdullah

Kelas: Gt-A11

Percobaan 1

Input:

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

int Data[MAX];

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

// Prosedur pengurutan metode gelembung
void BubbleSort()
{
    for (int i = 1; i < MAX - 1; i++)
    {
        for (int j = MAX - 1; j >= i; j--)
        {
            if (Data[j - 1] > Data[j])
            {
                Tukar(Data[j - 1], Data[j]);
            }
        }
    }
```

```
}

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;
    }

    BubbleSort();

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

    return 0;
}
```

Output:

```
/tmp/7eLzhGRx1f.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];

// Prosedur menukar data
void Tukar(int &a, int &b)
{
    int 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
    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;
    }

    ShellSort();

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

    return 0;
}
```

Output:

```
/tmp/7eLzhGRx1f.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 metode gelembung dan shell.

Input:

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

int Data[MAX];

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

// Prosedur pengurutan metode gelembung dengan tampilan setiap iterasi
void BubbleSort()
{
    std::cout << "Proses Pengurutan Metode Gelembung:" << std::endl;
    for (int i = 1; i < MAX - 1; i++)
    {
        for (int j = MAX - 1; j >= i; j--)
        {
            if (Data[j - 1] > Data[j])
            {
                Tukar(Data[j - 1], Data[j]);
                // Menampilkan perubahan setiap iterasi
                for (int k = 0; k < MAX; k++)
                {
```

```
                    std::cout << Data[k] << " ";
                }
                std::cout << std::endl;
            }
        }
    }
}

// Prosedur pengurutan metode Shell dengan tampilan setiap iterasi
void ShellSort()
{
    std::cout << "\nProses Pengurutan Metode Shell:" << std::endl;
    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;
                    // Menampilkan perubahan setiap iterasi
                    for (int k = 0; k < MAX; k++)
                    {
```

```
                        std::cout << Data[k] << " ";
                    }
                    std::cout << std::endl;
                }
            }
        }
    }
}

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;
    }

    BubbleSort();
    ShellSort();

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

```
    return 0;
}
```

Output:

```
/tmp/ciWYzHU1Nb.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
Proses Pengurutan Metode Gelembung:
384 887 778 916 794 336 387 493 422 650
384 887 778 916 794 336 387 422 493 650
384 887 778 916 336 794 387 422 493 650
384 887 778 336 916 794 387 422 493 650
384 887 336 778 916 794 387 422 493 650
384 336 887 778 916 794 387 422 493 650
336 384 887 778 916 794 387 422 493 650
336 384 887 778 916 387 794 422 493 650
336 384 887 778 387 916 794 422 493 650
336 384 887 387 778 916 794 422 493 650
336 384 387 887 778 916 794 422 493 650
336 384 387 887 778 916 422 794 493 650
336 384 387 887 778 422 916 794 493 650
336 384 387 887 422 778 916 794 493 650
336 384 387 422 887 778 916 794 493 650
336 384 387 422 887 778 916 493 794 650
336 384 387 422 887 778 493 916 794 650
336 384 387 422 887 493 778 916 794 650
336 384 387 422 493 887 778 916 794 650
336 384 387 422 493 887 778 916 650 794
336 384 387 422 493 887 778 650 916 794
336 384 387 422 493 887 650 778 916 794
336 384 387 422 493 650 887 778 916 794
336 384 387 422 493 650 887 778 794 916
336 384 387 422 493 650 778 887 794 916
336 384 387 422 493 650 778 794 887 916

Proses Pengurutan Metode Shell:

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 2

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

Input:

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

void bubbleSort(int arr[], int n, int &comparisons, int &shifts) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            comparisons++;
            if (arr[j] > arr[j+1]) {
                swap(arr[j], arr[j+1]);
                shifts++;
            }
        }
    }
}

void shellSort(int arr[], int n, int &comparisons, int &shifts) {
    for (int gap = n/2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i += 1) {
            int temp = arr[i];
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
                arr[j] = arr[j - gap];
                comparisons++;
                shifts++;
            }
            arr[j] = temp;
            if (i != j) shifts++;
        }
    }
```

```
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    int bubbleComparisons = 0, bubbleShifts = 0;
    int shellComparisons = 0, shellShifts = 0;

    // Bubble Sort
    bubbleSort(arr, n, bubbleComparisons, bubbleShifts);
    cout << "Bubble Sorted array: \n";
    for (int i=0; i < n; i++)
        cout << arr[i] << " ";
    cout << "\nJumlah perbandingan (Bubble Sort): " <<
bubbleComparisons << endl;
    cout << "Jumlah pergeseran (Bubble Sort): " << bubbleShifts <<
endl;

    // Reset array to unsorted state
    int arr2[] = {64, 34, 25, 12, 22, 11, 90};

    // Shell Sort
    shellSort(arr2, n, shellComparisons, shellShifts);
    cout << "\nShell Sorted array: \n";
    for (int i=0; i<n; i++)
        cout << arr2[i] << " ";
    cout << "\nJumlah perbandingan (Shell Sort): " <<
shellComparisons << endl;
    cout << "Jumlah pergeseran (Shell Sort): " << shellShifts <<
endl;

    return 0;
}
```

Output:

```
/tmp/QpYpOOnUQo.o
Bubble Sorted array:
11 12 22 25 34 64 90
Jumlah perbandingan (Bubble Sort): 21
Jumlah pergeseran (Bubble Sort): 14

Shell Sorted array:
11 12 22 25 34 64 90
Jumlah perbandingan (Shell Sort): 8
Jumlah pergeseran (Shell Sort): 14
```

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>
#include <functional>
using namespace std;

struct Pegawai {
    string NIP;
    string NAMA;
};

// Fungsi untuk membandingkan dua Pegawai berdasarkan NIP
```

```
bool compareByNIP(const Pegawai &a, const Pegawai &b) {
    return a.NIP < b.NIP; // Urut naik
}

// Fungsi untuk membandingkan dua Pegawai berdasarkan NAMA
bool compareByNAMA(const Pegawai &a, const Pegawai &b) {
    return a.NAMA < b.NAMA; // Urut naik
}

// Fungsi untuk mengurutkan array Pegawai
void sortPegawai(Pegawai arr[], int n, char metode, char urutan) {
    if (metode == 'N') {
        if (urutan == 'A') {
            sort(arr, arr + n, compareByNIP);
        } else {
            sort(arr, arr + n, [](const Pegawai &a, const Pegawai
&b) {
                return a.NIP > b.NIP; // Urut turun
            });
        }
    } else if (metode == 'M') {
        if (urutan == 'A') {
            sort(arr, arr + n, compareByNAMA);
        } else {
            sort(arr, arr + n, [](const Pegawai &a, const Pegawai
&b) {
                return a.NAMA > b.NAMA; // Urut turun
            });
        }
    }
}

int main() {
```

```
    Pegawai pegawai[] = {
        {"198203032003121001", "Dimas"},
        {"198103012003121002", "Elmo"},
        {"198303052003121003", "Toriq"},
        {"198503072003121004", "Jupri"}
    };
    int n = sizeof(pegawai) / sizeof(pegawai[0]);
    char metode, urutan;

    cout << "Pilih metode pengurutan (N untuk NIP, M untuk NAMA): ";
    cin >> metode;
    if (metode!= 'N' && metode!= 'M') {
        cout << "Metode pengurutan salah!" << endl;
        return -1;
    }

    cout << "Pilih urutan (A untuk naik, D untuk turun): ";
    cin >> urutan;
    if (urutan!= 'A' && urutan!= 'D') {
        cout << "Urutan salah!" << endl;
        return -1;
    }

    sortPegawai(pegawai, n, metode, urutan);

    cout << "Data Pegawai setelah diurutkan:\n";
    for (int i = 0; i < n; i++) {
        cout << "NIP: " << pegawai[i].NIP << ", NAMA: " <<
pegawai[i].NAMA << endl;
    }

    return 0;
}
```

Output:

```
/tmp/OZWjVKcoNx.o
Pilih metode pengurutan (N untuk NIP, M untuk NAMA): N
Pilih urutan (A untuk naik, D untuk turun): A
Data Pegawai setelah diurutkan:
NIP: 198103012003121002, NAMA: Elmo
NIP: 198203032003121001, NAMA: Dimas
NIP: 198303052003121003, NAMA: Toriq
NIP: 198503072003121004, NAMA: Jupri
```

```
/tmp/AcoEUIGOPC.o
Pilih metode pengurutan (N untuk NIP, M untuk NAMA): M
Pilih urutan (A untuk naik, D untuk turun): D
Data Pegawai setelah diurutkan:
NIP: 198303052003121003, NAMA: Toriq
NIP: 198503072003121004, NAMA: Jupri
NIP: 198103012003121002, NAMA: Elmo
NIP: 198203032003121001, NAMA: Dimas
```

Latihan 4

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

Pemahaman Algoritma Pengurutan: Praktikum ini memberikan pemahaman yang lebih dalam tentang algoritma pengurutan, khususnya Bubble Sort dan Shell Sort. Kita telah belajar bagaimana kedua algoritma ini bekerja dan bagaimana mereka dapat diimplementasikan dalam bahasa pemrograman C++.

Serta praktikum ini juga mengajarkan tentang penggunaan fungsi untuk memisahkan logika program dan struktur untuk mengelompokkan data terkait.