

Praktikum Algoritma dan Struktur Data

Bubble & Shell Sort



Oleh :

Rayhan Elmo Athalah Saputra (5223600027)

Program Studi Sarjana Terapan Teknologi Game

Departemen Teknologi Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2024

A. Percobaan

1. Pengurutan dengan metode Bubble sort.

[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 = 0; j < MAX - i - 1; j++) { // Ubah perulangan j
            // Membandingkan dua elemen berturut-turut dan menukar
            // posisi jika perlu
            if (Data[j] > Data[j + 1]) { // Ubah tanda perbandingan
                Tukar(&Data[j], &Data[j + 1]);
            }
        }
    }
}

int main()
{
    srand(0);

    // Membangkitkan bilangan acak
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++) {
        Data[i] = rand() % 100 + 1; // Batasi angka yang dibangkitkan
```

```

        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;
}

```

The screenshot shows the Programiz C++ Online Compiler interface. The code in the editor is as follows:

```

1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6
7 using namespace std;
8
9 // Prosedur menukar data
10 void Tukar(int* a, int* b)
11 {
12     int temp;
13     temp = *a;
14     *a = *b;
15     *b = temp;
16 }
17
18 // Prosedur pengurutan metode gelembung
19 void BubbleSort()
20 {
21     for (int i = 0; i < MAX - 1; i++) {

```

The output window shows the following results:

```

/tmp/IUmImcQR7x.o
DATA SEBELUM TERURUT
Data ke 0 : 84
Data ke 1 : 87
Data ke 2 : 78
Data ke 3 : 16
Data ke 4 : 94
Data ke 5 : 36
Data ke 6 : 87
Data ke 7 : 93
Data ke 8 : 50
Data ke 9 : 22

DATA SETELAH TERURUT
Data ke 0 : 16
Data ke 1 : 22
Data ke 2 : 36
Data ke 3 : 50
Data ke 4 : 78
Data ke 5 : 84
Data ke 6 : 87

```

Output

2. Pengurutan dengan metode Shell sort.

[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 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() % 100 + 1; // Batasi angka yang
dibangkitkan
        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;
}

```

The screenshot shows the Programiz C++ Online Compiler interface. On the left, the code for `main.cpp` is displayed, including headers, a constant `MAX` of 10, a swap function, and the `ShellSort` function. On the right, the 'Output' pane shows the program's execution results. It first displays 'DATA SEBELUM TERURUT' followed by 10 lines of data. Then it displays 'DATA SETELAH TERURUT' followed by the same 10 lines of data, now sorted in ascending order.

```

main.cpp
1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6
7 using namespace std;
8
9 // Prosedur menukar data
10 void Tukar(int* a, int* b)
11 {
12     int temp;
13     temp = *a;
14     *a = *b;
15     *b = temp;
16 }
17
18 // Prosedur pengurutan metode Shell
19 void ShellSort()
20 {
21     int Jarak, i, j;
22     // ...

```

Output

```

/tmp/C8TjinLf0.o
DATA SEBELUM TERURUT
Data ke 0 : 84
Data ke 1 : 87
Data ke 2 : 78
Data ke 3 : 16
Data ke 4 : 94
Data ke 5 : 36
Data ke 6 : 87
Data ke 7 : 93
Data ke 8 : 50
Data ke 9 : 22

DATA SETELAH TERURUT
Data ke 0 : 16
Data ke 1 : 22
Data ke 2 : 36
Data ke 3 : 50
Data ke 4 : 78
Data ke 5 : 84
Data ke 6 : 87

```

Output

B. Latihan

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() % 100 + 1; // Batasi angka yang
dibangkitkan
        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;
}

```

The screenshot shows the Programiz C++ Online Compiler interface. The code in the editor is as follows:

```

1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6
7 using namespace std;
8
9 // Prosedur menukar data
10 void Tukar(int* a, int* b)
11 {
12     int temp;
13     temp = *a;
14     *a = *b;
15     *b = temp;
16 }
17
18 // Prosedur pengurutan metode Bubble Sort
19 void BubbleSort()
20 {
21     cout << "\n\nBUBBLE SORT PROCESS:" << endl;
22 }

```

The output window shows the following results:

```

/tmp/BZyztJFr24.o
DATA SEBELUM TERURUT
Data ke 0 : 84
Data ke 1 : 87
Data ke 2 : 78
Data ke 3 : 16
Data ke 4 : 94
Data ke 5 : 36
Data ke 6 : 87
Data ke 7 : 93
Data ke 8 : 50
Data ke 9 : 22

BUBBLE SORT PROCESS:
Iteration 1: 84 78 16 87 36 87 93 50 22 94
Iteration 2: 78 16 84 36 87 87 50 22 93 94
Iteration 3: 16 78 36 84 87 50 22 87 93 94
Iteration 4: 16 36 78 84 50 22 87 87 93 94
Iteration 5: 16 36 78 50 22 84 87 87 93 94
Iteration 6: 16 36 50 22 78 84 87 87 93 94

```

Output

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() % 100 + 1; // Batasi angka yang
        dibangkitkan
        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;
}

```

Programiz C++ Online Compiler

main.cpp

```

1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6 int comparisonCountBubble = 0;
7 int shiftCountBubble = 0;
8 int comparisonCountShell = 0;
9 int shiftCountShell = 0;
10
11 using namespace std;
12
13 // Prosedur menukar data
14 void Tukar(int* a, int* b)
15 {
16     int temp;
17     temp = *a;
18     *a = *b;
19     *b = temp;
20 }
21
22 // Prosedur mengurutkan data dengan Bubble Sort
23 void UrutkanBubble(int Data[])
24 {
25     for (int i = 0; i < MAX - 1; i++)
26     {
27         for (int j = 0; j < MAX - i - 1; j++)
28         {
29             if (Data[j] > Data[j + 1])
30             {
31                 Tukar(&Data[j], &Data[j + 1]);
32             }
33             comparisonCountBubble++;
34             shiftCountBubble++;
35         }
36     }
37 }
38
39 int main()
40 {
41     Data[0] = 84;
42     Data[1] = 78;
43     Data[2] = 16;
44     Data[3] = 87;
45     Data[4] = 36;
46     Data[5] = 94;
47     Data[6] = 87;
48     Data[7] = 93;
49     Data[8] = 50;
50     Data[9] = 22;
51
52     UrutkanBubble(Data);
53
54     return 0;
55 }

```

Output

```

/tmp/o5HLUp3wh.o
DATA SEBELUM TERURUT
Data ke 0 : 84
Data ke 1 : 78
Data ke 2 : 16
Data ke 3 : 87
Data ke 4 : 36
Data ke 5 : 94
Data ke 6 : 87
Data ke 7 : 93
Data ke 8 : 50
Data ke 9 : 22

BUBBLE SORT PROCESS:
Iteration 1: 84 78 16 87 36 87 93 50 22 94
Iteration 2: 78 16 84 36 87 87 50 22 93 94
Iteration 3: 16 78 36 84 87 50 22 87 93 94
Iteration 4: 16 36 78 84 50 22 87 87 93 94
Iteration 5: 16 36 78 50 22 84 87 87 93 94
Iteration 6: 16 36 50 22 78 84 87 87 93 94

```

Programiz C++ Online Compiler

main.cpp

```

1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6 int comparisonCountBubble = 0;
7 int shiftCountBubble = 0;
8 int comparisonCountShell = 0;
9 int shiftCountShell = 0;
10
11 using namespace std;
12
13 // Prosedur menukar data
14 void Tukar(int* a, int* b)
15 {
16     int temp;
17     temp = *a;
18     *a = *b;
19     *b = temp;
20 }
21
22 // Prosedur mengurutkan data dengan Shell Sort
23 void UrutkanShell(int Data[])
24 {
25     int gap = MAX / 2;
26     while (gap > 0)
27     {
28         for (int i = gap; i < MAX; i++)
29         {
30             for (int j = i - gap; j >= 0; j -= gap)
31             {
32                 if (Data[j] > Data[j + gap])
33                 {
34                     Tukar(&Data[j], &Data[j + gap]);
35                 }
36                 comparisonCountShell++;
37                 shiftCountShell++;
38             }
39         }
40         gap = gap / 2;
41     }
42 }
43
44 int main()
45 {
46     Data[0] = 16;
47     Data[1] = 22;
48     Data[2] = 36;
49     Data[3] = 50;
50     Data[4] = 78;
51     Data[5] = 84;
52     Data[6] = 87;
53     Data[7] = 87;
54     Data[8] = 93;
55     Data[9] = 94;
56
57     UrutkanShell(Data);
58
59     return 0;
60 }

```

Output

```

SHELL SORT PROCESS:
Gap 5: 16 22 36 50 78 84 87 87 93 94
Gap 2: 16 22 36 50 78 84 87 87 93 94
Gap 1: 16 22 36 50 78 84 87 87 93 94

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

DATA SETELAH TERURUT
Data ke 0 : 16
Data ke 1 : 22
Data ke 2 : 36
Data ke 3 : 50
Data ke 4 : 78
Data ke 5 : 84
Data ke 6 : 87
Data ke 7 : 87
Data ke 8 : 93
Data ke 9 : 94

```

Output

3. Tambahkan pada project Latihan pada praktikum 7 dan implementasikan pengurutan data Pegawai pada tugas pendahuluan dengan ketentuan :

- **Metode pengurutan dapat dipilih.**
- **Pengurutan dapat dipilih secara urut naik atau turun.**
- **Pengurutan dapat dipilih berdasarkan NIP dan NAMA.**
- **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;
    string JENIS_KELAMIN;
};

// 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] = {
        {"027", "Rayhan Elmo", "Laki-laki"},
        {"006", "Fina Salsa", "Perempuan"},
        {"001", "Oscar Javier", "Laki-laki"},
        {"004", "Hatfan Sahrul", "Laki-laki"},
        {"011", "Tiyassha Ananda", "Perempuan"},
        {"013", "Tora Kamulian", "Laki-laki"},
        {"024", "Dio Standia", "Laki-laki"},
        {"028", "Rida Handayani", "Perempuan"},
        {"030", "Fahrial Ananta", "Laki-laki"},
        {"002", "Bernadus Hendry", "Laki-laki"},

    };

    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 pembandingan 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;
        }
    }
}

```

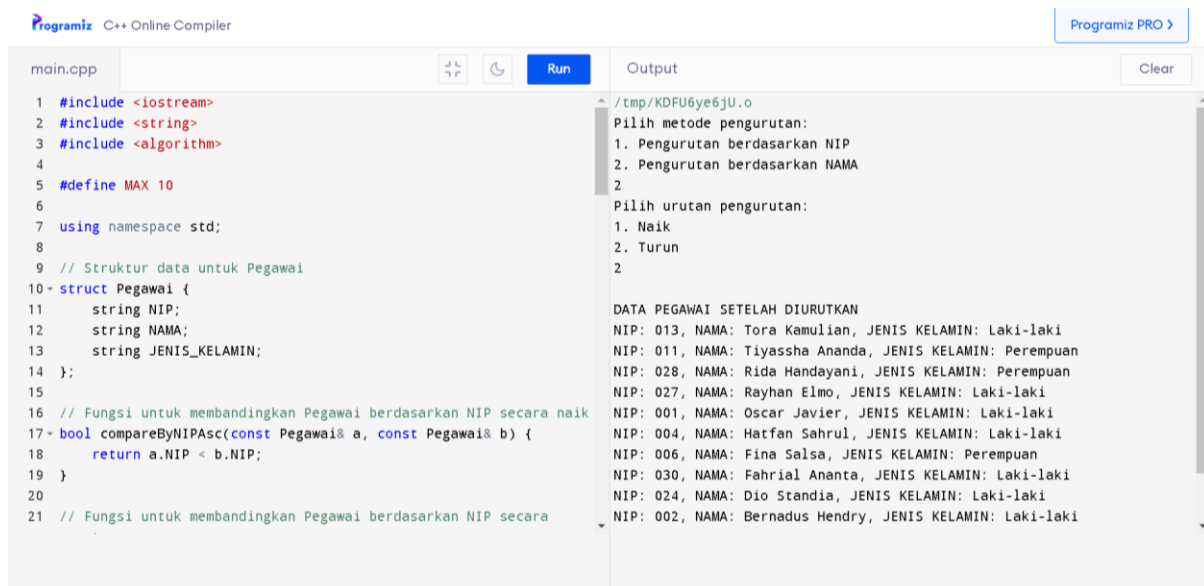
```

// Melakukan pengurutan menggunakan fungsi pembandingan 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 << ", JENIS KELAMIN: " <<
dataPegawai[i].JENIS_KELAMIN << endl;
}

return 0;
}

```



The screenshot shows a C++ Online Compiler interface. On the left, the source code for 'main.cpp' is displayed, featuring includes for iostream, string, and algorithm, a MAX constant of 10, a Pegawai struct, and sorting functions. On the right, the 'Output' window shows the program's execution results, including menu options for sorting by NIP or Name, and a list of sorted employee data.

```

main.cpp
1 #include <iostream>
2 #include <string>
3 #include <algorithm>
4
5 #define MAX 10
6
7 using namespace std;
8
9 // Struktur data untuk Pegawai
10 struct Pegawai {
11     string NIP;
12     string NAMA;
13     string JENIS_KELAMIN;
14 };
15
16 // Fungsi untuk membandingkan Pegawai berdasarkan NIP secara naik
17 bool compareByNIPasc(const Pegawai& a, const Pegawai& b) {
18     return a.NIP < b.NIP;
19 }
20
21 // Fungsi untuk membandingkan Pegawai berdasarkan NIP secara

```

```

/tmp/KDFU6ye6jU.o
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: 013, NAMA: Tora Kamulian, JENIS KELAMIN: Laki-laki
NIP: 011, NAMA: Tiyyasha Ananda, JENIS KELAMIN: Perempuan
NIP: 028, NAMA: Rida Handayani, JENIS KELAMIN: Perempuan
NIP: 027, NAMA: Rayhan Elmo, JENIS KELAMIN: Laki-laki
NIP: 001, NAMA: Oscar Javier, JENIS KELAMIN: Laki-laki
NIP: 004, NAMA: Hatfan Sahrul, JENIS KELAMIN: Laki-laki
NIP: 006, NAMA: Fina Salsa, JENIS KELAMIN: Perempuan
NIP: 030, NAMA: Fahrial Ananta, JENIS KELAMIN: Laki-laki
NIP: 024, NAMA: Dio Standia, JENIS KELAMIN: Laki-laki
NIP: 002, NAMA: Bernadus Hendry, JENIS KELAMIN: Laki-laki

```

Output

Kesimpulan

Dalam seri program yang telah dijelaskan, kami menerapkan dan memodifikasi algoritma Bubble Sort dan Shell Sort dalam bahasa C++. Bubble Sort adalah metode yang mudah dimengerti di mana elemen-elemen dalam array dibandingkan dan ditukar berulang kali sampai array tersebut terurut. Namun, Bubble Sort cenderung kurang efisien untuk data besar karena kompleksitas waktu terburuknya adalah $O(n^2)$.

Di sisi lain, Shell Sort merupakan pengembangan dari Insertion Sort yang mengurutkan elemen dalam interval tertentu sebelum akhirnya menggunakan Insertion Sort konvensional.

Dengan memungkinkan pertukaran elemen yang berjauhan, Shell Sort dapat menghasilkan pengurutan lebih cepat, terutama pada data besar. Shell Sort memiliki kompleksitas waktu yang bervariasi tergantung pada interval yang dipilih, dan umumnya lebih efisien daripada Bubble Sort.

Implementasi dan perbandingan kedua algoritma ini dalam konteks pengurutan data karyawan memberikan pemahaman praktis tentang pemilihan algoritma yang sesuai untuk menangani masalah pengurutan data dengan efisiensi dan efektivitas yang maksimal.