

Praktikum Algoritma dan Struktur Data

Insertion & Selection 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 penyisipan langsung (straight insertion sort)

[Input]

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

#define MAX 5
int Data[MAX];

using namespace std;

// Function for straight insertion sort
void StraightInsertSort()
{
    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--;
        }
        Data[j + 1] = x;
    }
}

int main()
{
    srand(0);

    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = (rand() % 100) + 1; // Angka antara 1 dan 100
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }

    StraightInsertSort();

    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 editor on the left contains a C++ program for straight insertion sort. The output window on the right displays the results of the program execution.

```

main.cpp
1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 5
5 int Data[MAX];
6
7 using namespace std;
8
9 // Function for straight insertion sort
10 void StraightInsertSort()
11 {
12     for (int i = 1; i < MAX; i++) {
13         int x = Data[i];
14         int j = i - 1;
15         while (j >= 0 && x < Data[j]) {
16             Data[j + 1] = Data[j];
17             j--;
18         }
19         Data[j + 1] = x;
20     }
21 }
22

```

Output

```

/tmp/SHTzWRCXD0.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 SETELAH TERURUT
Data ke 0 : 16
Data ke 1 : 78
Data ke 2 : 84
Data ke 3 : 87
Data ke 4 : 94

=== Code Execution Successful ===

```

Output

2. Pengurutan dengan metode penyisipan biner (binary insertion sort)

[Input]

```

#include <iostream>
#include <cstdlib>

#define MAX 10
int Data[MAX];

using namespace std;

void BinaryInsertSort()
{
    for (int i = 1; i < MAX; i++) {
        int x = Data[i];
        int l = 0;
        int r = i - 1;
        while (l <= r) {
            int m = (l + r) / 2;

```

```

        if (x < Data[m])
            r = m - 1;
        else
            l = m + 1;
    }
    for (int j = i - 1; j >= l; j--)
        Data[j + 1] = Data[j];
    Data[l] = x;
}
}

int main()
{
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = (rand() % 100) + 1; // Angka antara 1 dan 100
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    BinaryInsertSort();
    cout << "\nDATA SETELAH TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}

```

The screenshot shows a C++ program being executed on an online compiler. The code defines a constant MAX of 10, generates 10 random numbers, and sorts them using a Binary Insert Sort algorithm. The output displays the data before and after sorting.

```

main.cpp
1  #include <iostream>
2  #include <cstdlib>
3
4  #define MAX 10
5  int Data[MAX];
6
7  using namespace std;
8
9  void BinaryInsertSort()
10 {
11     for (int i = 1; i < MAX; i++) {
12         int x = Data[i];
13         int l = 0;
14         int r = i - 1;
15         while (l <= r) {
16             int m = (l + r) / 2;
17             if (x < Data[m])
18                 r = m - 1;
19             else
20                 l = m + 1;
21         }
22         for (int j = i - 1; j >= l; j--)
23             Data[j + 1] = Data[j];
24         Data[l] = x;
25     }
26 }

```

Output

```

/tmp/UGuf20ZAw1.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

3. Pengurutan dengan metode seleksi (selection sort)

[Input]

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

int main()
{
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = rand() % 100; // Angka antara 0 dan 99
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    SelectionSort();
    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 left pane displays the source code for `main.cpp`, and the right pane shows the program's output.

Source Code (main.cpp):

```

1  #include <iostream>
2  #include <cstdlib>
3
4  #define MAX 10
5  int Data[MAX];
6
7  using namespace std;
8
9  void Tukar(int* a, int* b)
10 {
11     int temp;
12     temp = *a;
13     *a = *b;
14     *b = temp;
15 }
16
17 void SelectionSort()
18 {
19     for (int i = 0; i < MAX - 1; i++)
20     {
21         int k = i;
22         for (int j = i + 1; j < MAX; j++)
23             if (Data[j] < Data[k])
24                 k = j;
25         Tukar(&Data[i], &Data[k]);
26     }
27 }
28
29 int main()
30 {
31     int i;
32     for (i = 0; i < MAX; i++)
33         Data[i] = rand() % 100;
34     cout << "DATA SEBELUM TERURUT\n";
35     for (i = 0; i < MAX; i++)
36         cout << "Data ke " << i << " : " << Data[i] << "\n";
37     SelectionSort();
38     cout << "DATA SETELAH TERURUT\n";
39     for (i = 0; i < MAX; i++)
40         cout << "Data ke " << i << " : " << Data[i] << "\n";
41     return 0;
42 }

```

Output:

```

/tmp/iUSKYs0Saf.o
DATA SEBELUM TERURUT
Data ke 0 : 83
Data ke 1 : 86
Data ke 2 : 77
Data ke 3 : 15
Data ke 4 : 93
Data ke 5 : 35
Data ke 6 : 86
Data ke 7 : 92
Data ke 8 : 49
Data ke 9 : 21
DATA SETELAH TERURUT
Data ke 0 : 15
Data ke 1 : 21
Data ke 2 : 35
Data ke 3 : 49
Data ke 4 : 77
Data ke 5 : 83
Data ke 6 : 86
Data ke 7 : 92
Data ke 8 : 93
Data ke 9 : 86

```

Output

B. Latihan

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];

using namespace std;

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--;
        }
        Data[j + 1] = x;

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

void SelectionSort()
{
    cout << "\n\nPROSES PENYORTIRAN DENGAN SELEKSI:" << endl;

    for (int i = 0; i < MAX - 1; i++)
    {
        int k = i;
        for (int j = i + 1; j < MAX; j++)
            if (Data[k] > Data[j])
                k = j;

        Tukar(&Data[i], &Data[k]);

        cout << "Iterasi " << i + 1 << ": ";
        for (int k = 0; k < MAX; k++) {
            cout << Data[k] << " ";
        }
        cout << endl;
    }
}

```

```

void BinaryInsertSort()
{
    cout << "\n\nPROSES PENYORTIRAN DENGAN BINARY INSERTION:" <<
endl;

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

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

int main()
{
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = rand() % 100; // Angka antara 0 dan 99
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }

    StraightInsertSort();
    SelectionSort();
    BinaryInsertSort();

    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 implements a Straight Insertion Sort. The output window displays the initial data, the sorting process with iterations, and the final sorted data.

```

main.cpp
1 #include <iostream>
2 #include <cstdlib>
3
4 #define MAX 10
5 int Data[MAX];
6
7 using namespace std;
8
9 void Tukar(int* a, int* b)
10 {
11     int temp;
12     temp = *a;
13     *a = *b;
14     *b = temp;
15 }
16
17 void StraightInsertSort()
18 {
19     cout << "\n\nPROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:" << endl;
20
21     for (int i = 1; i < MAX; i++) {
22         int x = Data[i];
23         int j = i - 1;
24         while (j >= 0 && x < Data[j]) {
25             Data[j+1] = Data[j];

```

Output

```

/tmp/qFuUuDVJry.o
DATA SEBELUM TERURUT
Data ke 0 : 83
Data ke 1 : 86
Data ke 2 : 77
Data ke 3 : 15
Data ke 4 : 93
Data ke 5 : 35
Data ke 6 : 86
Data ke 7 : 92
Data ke 8 : 49
Data ke 9 : 21

PROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:
Iterasi 1: 83 86 77 15 93 35 86 92 49 21
Iterasi 2: 77 83 86 15 93 35 86 92 49 21
Iterasi 3: 15 77 83 86 93 35 86 92 49 21
Iterasi 4: 15 77 83 86 93 35 86 92 49 21
Iterasi 5: 15 35 77 83 86 93 86 92 49 21
Iterasi 6: 15 35 77 83 86 86 93 92 49 21
Iterasi 7: 15 35 77 83 86 86 92 93 49 21
Iterasi 8: 15 35 49 77 83 86 86 92 93 21
Iterasi 9: 15 21 35 49 77 83 86 86 92 93

```

Output Straight Insertion

The screenshot shows the Programiz C++ Online Compiler interface. The code in the editor implements a Selection Sort. The output window displays the sorting process with iterations and the final sorted data.

```

main.cpp
37 void SelectionSort()
38 {
39     cout << "\n\nPROSES PENYORTIRAN DENGAN SELEKSI:" << endl;
40
41     for (int i = 0; i < MAX - 1; i++)
42     {
43         int k = i;
44         for (int j = i + 1; j < MAX; j++)
45             if (Data[k] > Data[j])
46                 k = j;
47
48         Tukar(&Data[i], &Data[k]);
49
50         cout << "Iterasi " << i + 1 << " : ";
51         for (int k = 0; k < MAX; k++) {
52             cout << Data[k] << " ";
53         }
54         cout << endl;
55     }
56 }
57
58 int main()
59 {
60     srand(0);

```

Output

```

PROSES PENYORTIRAN DENGAN SELEKSI:
Iterasi 1: 15 21 35 49 77 83 86 86 92 93
Iterasi 2: 15 21 35 49 77 83 86 86 92 93
Iterasi 3: 15 21 35 49 77 83 86 86 92 93
Iterasi 4: 15 21 35 49 77 83 86 86 92 93
Iterasi 5: 15 21 35 49 77 83 86 86 92 93
Iterasi 6: 15 21 35 49 77 83 86 86 92 93
Iterasi 7: 15 21 35 49 77 83 86 86 92 93
Iterasi 8: 15 21 35 49 77 83 86 86 92 93
Iterasi 9: 15 21 35 49 77 83 86 86 92 93

DATA SETELAH TERURUT
Data ke 0 : 15
Data ke 1 : 21
Data ke 2 : 35
Data ke 3 : 49
Data ke 4 : 77
Data ke 5 : 83
Data ke 6 : 86
Data ke 7 : 86
Data ke 8 : 92
Data ke 9 : 93

```

Ouput Selection Sort

Programiz
C++ Online Compiler
Programiz PRO >

main.cpp

Run

Output

Clear

```

58
59 void BinaryInsertSort()
60 {
61     cout << "\n\nPROSES PENYORTIRAN DENGAN BINARY INSERTION:" << endl;
62
63     for (int i = 1; i < MAX; i++) {
64         int x = Data[i];
65         int l = 0;
66         int r = i - 1;
67         while (l <= r) {
68             int m = (l + r) / 2;
69             if (x < Data[m])
70                 r = m - 1;
71             else
72                 l = m + 1;
73         }
74         for (int j = i - 1; j >= l; j--)
75             Data[j + 1] = Data[j];
76         Data[l + 1] = x;
77
78         cout << "Iterasi " << i << " : ";
79         for (int k = 0; k < MAX; k++) {
80             cout << Data[k] << " ";
81         }
82         cout << endl;

```

PROSES PENYORTIRAN DENGAN BINARY INSERTION:

Iterasi 1: 15 21 35 49 77 83 86 86 92 93
Iterasi 2: 15 21 35 49 77 83 86 86 92 93
Iterasi 3: 15 21 35 49 77 83 86 86 92 93
Iterasi 4: 15 21 35 49 77 83 86 86 92 93
Iterasi 5: 15 21 35 49 77 83 86 86 92 93
Iterasi 6: 15 21 35 49 77 83 86 86 92 93
Iterasi 7: 15 21 35 49 77 83 86 86 92 93
Iterasi 8: 15 21 35 49 77 83 86 86 92 93
Iterasi 9: 15 21 35 49 77 83 86 86 92 93

DATA SETELAH TERURUT

Data ke 0 : 15
Data ke 1 : 21
Data ke 2 : 35
Data ke 3 : 49
Data ke 4 : 77
Data ke 5 : 83
Data ke 6 : 86
Data ke 7 : 86
Data ke 8 : 92
Data ke 9 : 93

Output Binary Insertion

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];

using namespace std;

int comparisonCount = 0;
int shiftCount = 0;

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++;
            shiftCount++;
        }
        Data[j + 1] = x;
        shiftCount++;

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

        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;
    }

void BinaryInsertSort()
{
    cout << "\n\nPROSES PENYORTIRAN DENGAN BINARY INSERTION:" <<
endl;
    comparisonCount = 0;
    shiftCount = 0;
    for (int i = 1; i < MAX; i++) {
        int x = Data[i];
        int l = 0;
        int r = i - 1;
        while (l <= r) {
            int m = (l + r) / 2;
            comparisonCount++;
            if (x < Data[m])
                r = m - 1;
            else
                l = m + 1;
        }
        for (int j = i - 1; j >= l; j--)
            Data[j + 1] = Data[j];
        Data[l] = x;
        shiftCount++;

        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;
}

int main()
{
    srand(0);
    cout << "DATA SEBELUM TERURUT" << endl;
    for (int i = 0; i < MAX; i++)
    {
        Data[i] = rand() % 100; // Menghasilkan angka antara 0 dan 99
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }

    StraightInsertSort();
}

```

```

SelectionSort();

BinaryInsertSort();

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

```

main.cpp

```

19
20 void StraightInsertSort()
21 {
22     cout << "\n\nPROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:" << endl;
23     for (int i = 1; i < MAX; i++) {
24         int x = Data[i];
25         int j = i - 1;
26         while (j >= 0 && x < Data[j]) {
27             Data[j + 1] = Data[j];
28             j--;
29             comparisonCount++;
30             shiftCount++;
31         }
32         Data[j + 1] = x;
33         shiftCount++;
34
35         cout << "Iterasi " << i << " : ";
36         for (int k = 0; k < MAX; k++) {
37             cout << Data[k] << " ";
38         }
39         cout << endl;
40     }
41     cout << "Jumlah perbandingan: " << comparisonCount << endl;
42     cout << "Jumlah pergeseran: " << shiftCount << endl;
43 }

```

Output

```

DATA SEBELUM TERURUT
Data ke 0 : 83
Data ke 1 : 86
Data ke 2 : 77
Data ke 3 : 15
Data ke 4 : 93
Data ke 5 : 35
Data ke 6 : 86
Data ke 7 : 92
Data ke 8 : 49
Data ke 9 : 21

PROSES PENYORTIRAN DENGAN PENYISIPAN LANGSUNG:
Iterasi 1: 83 86 77 15 93 35 86 92 49 21
Iterasi 2: 77 83 86 15 93 35 86 92 49 21
Iterasi 3: 15 77 83 86 93 35 86 92 49 21
Iterasi 4: 15 77 83 86 93 35 86 92 49 21
Iterasi 5: 15 35 77 83 86 93 86 92 49 21
Iterasi 6: 15 35 77 83 86 86 93 92 49 21
Iterasi 7: 15 35 77 83 86 86 92 93 49 21
Iterasi 8: 15 35 49 77 83 86 86 92 93 21
Iterasi 9: 15 21 35 49 77 83 86 86 92 93
Jumlah perbandingan: 25
Jumlah pergeseran: 34

```

Output Straight Inserstion

main.cpp

```

44
45 void SelectionSort()
46 {
47     cout << "\n\nPROSES PENYORTIRAN DENGAN SELEKSI:" << endl;
48     comparisonCount = 0;
49     shiftCount = 0;
50     for (int i = 0; i < MAX - 1; i++)
51     {
52         int k = i;
53         for (int j = i + 1; j < MAX; j++) {
54             comparisonCount++;
55             if (Data[k] > Data[j])
56                 k = j;
57         }
58         Tukar(&Data[i], &Data[k]);
59         shiftCount++;
60
61         cout << "Iterasi " << i + 1 << " : ";
62         for (int k = 0; k < MAX; k++) {
63             cout << Data[k] << " ";
64         }

```

Output

```

PROSES PENYORTIRAN DENGAN SELEKSI:
Iterasi 1: 15 21 35 49 77 83 86 86 92 93
Iterasi 2: 15 21 35 49 77 83 86 86 92 93
Iterasi 3: 15 21 35 49 77 83 86 86 92 93
Iterasi 4: 15 21 35 49 77 83 86 86 92 93
Iterasi 5: 15 21 35 49 77 83 86 86 92 93
Iterasi 6: 15 21 35 49 77 83 86 86 92 93
Iterasi 7: 15 21 35 49 77 83 86 86 92 93
Iterasi 8: 15 21 35 49 77 83 86 86 92 93
Iterasi 9: 15 21 35 49 77 83 86 86 92 93
Jumlah perbandingan: 45
Jumlah pergeseran: 9

PROSES PENYORTIRAN DENGAN BINARY INSERTION:
Iterasi 1: 15 21 35 49 77 83 86 86 92 93
Iterasi 2: 15 21 35 49 77 83 86 86 92 93
Iterasi 3: 15 21 35 49 77 83 86 86 92 93
Iterasi 4: 15 21 35 49 77 83 86 86 92 93

```

Output Selection Sort

```
main.cpp
71 void BinaryInsertSort()
72 {
73     cout << "\n\nPROSES PENYORTIRAN DENGAN BINARY INSERTION:" << endl;
74     comparisonCount = 0;
75     shiftCount = 0;
76     for (int i = 1; i < MAX; i++) {
77         int x = Data[i];
78         int l = 0;
79         int r = i - 1;
80         while (l <= r) {
81             int m = (l + r) / 2;
82             comparisonCount++;
83             if (x < Data[m])
84                 r = m - 1;
85             else
86                 l = m + 1;
87         }
88         for (int j = i - 1; j >= l; j--)
89             Data[j + 1] = Data[j];
90         Data[l + 1] = x;
91         shiftCount++;
92     }
93     cout << "Iterasi " << i << " : ";
94     for (int k = 0; k < MAX; k++) {
95         cout << Data[k] << " ";
96     }
97     cout << endl;
98 }

Output
PROSES PENYORTIRAN DENGAN BINARY INSERTION:
Iterasi 1: 15 21 35 49 77 83 86 86 92 93
Iterasi 2: 15 21 35 49 77 83 86 86 92 93
Iterasi 3: 15 21 35 49 77 83 86 86 92 93
Iterasi 4: 15 21 35 49 77 83 86 86 92 93
Iterasi 5: 15 21 35 49 77 83 86 86 92 93
Iterasi 6: 15 21 35 49 77 83 86 86 92 93
Iterasi 7: 15 21 35 49 77 83 86 86 92 93
Iterasi 8: 15 21 35 49 77 83 86 86 92 93
Iterasi 9: 15 21 35 49 77 83 86 86 92 93
Jumlah perbandingan: 25
Jumlah pergeseran: 9

DATA SETELAH TERURUT
Data ke 0 : 15
Data ke 1 : 21
Data ke 2 : 35
Data ke 3 : 49
Data ke 4 : 77
Data ke 5 : 83
Data ke 6 : 86
Data ke 7 : 86
Data ke 8 : 92
Data ke 9 : 93
```

Output Binary Insertion

3. Buatlah project baru untuk Latihan 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 <algorithm>
#include <string>
```

```
using namespace std;
```

```
// Struktur data untuk Pegawai
```

```
struct Pegawai {
    int NIP; // Nomor Induk Pegawai
    string Nama; // Nama pegawai
    char Gender; // Gender pegawai: 'L' (Laki-laki) atau 'P' (Perempuan)
};
```

```
// Fungsi untuk mencetak daftar Pegawai
```

```

void cetakPegawai(const Pegawai pegawai[], int ukuran) {
    for (int i = 0; i < ukuran; i++) {
        cout << "NIP: " << pegawai[i].NIP << ", Nama: " <<
pegawai[i].Nama << ", Gender: " << pegawai[i].Gender << endl;
    }
    cout << "-----" << endl; // Separator
}

// Implementasi pengurutan data Pegawai dengan Binary Insertion Sort
berdasarkan NIP secaraurut naik
void BinaryInsertionSort_NIP(Pegawai arr[], int n, bool naik) {
    for (int i = 1; i < n; i++) {
        Pegawai key = arr[i];
        int j = i - 1;
        while (j >= 0 && ((naik && arr[j].NIP > key.NIP) || (!naik &&
arr[j].NIP < key.NIP))) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// Implementasi pengurutan data Pegawai dengan Binary Insertion Sort
berdasarkan Nama secaraurut naik
void BinaryInsertionSort_Nama(Pegawai arr[], int n, bool naik) {
    for (int i = 1; i < n; i++) {
        Pegawai key = arr[i];
        int j = i - 1;
        while (j >= 0 && ((naik && arr[j].Nama > key.Nama) || (!naik
&& arr[j].Nama < key.Nama))) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

int main() {
    const int jumlahPegawai = 6; // Jumlah pegawai dalam array
    Pegawai pegawai[jumlahPegawai] = {
        {027, "Rayhan ELmo", 'L'},
        {001, "Oscar Javier", 'L'},
        {006, "Fina Salsa", 'P'},
        {013, "Tora Kamulian", 'L'},
        {004, "Hatfan Sahrul", 'L'},
        {011, "Tiyasha Ananda", 'P'}
    }
}

```

```

};

cout << "Daftar Pegawai Sebelum Terurut:\n";
cetakPegawai(pegawai, jumlahPegawai);

// Pilih metode pengurutan
char pilihanMetode = '\0';
cout << "Pilih Metode Pengurutan:\n";
cout << "a. Berdasarkan NIP\nb. Berdasarkan Nama\n";
cout << "Masukkan pilihan: ";
cin >> pilihanMetode;

// Pilih urutan naik atau turun
char pilihanUrutan = '\0';
cout << "Pilih Urutan:\n";
cout << "a. Naik\nb. Turun\n";
cout << "Masukkan pilihan: ";
cin >> pilihanUrutan;

bool naik = (pilihanUrutan == 'a');

// Pengurutan berdasarkan pilihan
if (pilihanMetode == 'a') { // Pengurutan berdasarkan NIP
    BinaryInsertionSort_NIP(pegawai, jumlahPegawai, naik);
}
else if (pilihanMetode == 'b') { // Pengurutan berdasarkan Nama
    BinaryInsertionSort_Nama(pegawai, jumlahPegawai, naik);
}
else {
    cout << "Pilihan tidak valid." << endl; // Kesalahan input
    return 1; // Menghentikan program dengan kesalahan
}

cout << "\nDaftar Pegawai Setelah Terurut:\n";
cetakPegawai(pegawai, jumlahPegawai);

return 0;
}

```



```
1 #include <iostream>
2 #include <algorithm>
3 #include <string>
4
5 using namespace std;
6
7 // Struktur data untuk Pegawai
8 struct Pegawai {
9     int NIP; // Nomor Induk Pegawai
10    string Nama; // Nama pegawai
11    char Gender; // Gender pegawai: 'L' (Laki-laki) atau 'P' (Perempuan)
12 };
13
14 // Fungsi untuk mencetak daftar Pegawai
15 void cetakPegawai(const Pegawai pegawai[], int ukuran) {
16     for (int i = 0; i < ukuran; i++) {
17         cout << "NIP: " << pegawai[i].NIP << ", Nama: " << pegawai[i].Nama << ",
18             Gender: " << pegawai[i].Gender << endl;
19     }
20     cout << "-----" << endl; // Separator
21 }
22
23 // Implementasi pengurutan data Pegawai dengan Binary Insertion Sort berdasarkan
24 // NIP secara urut naik
25 void BinaryInsertionSort(Pegawai arr[], int n, bool naik) {
26     for (int i = 1; i < n; i++) {
27         Pegawai key = arr[i];
28         int j = i - 1;
29         while (j >= 0 && ((naik && arr[j].NIP > key.NIP) || (!naik && arr[j].NIP < key.NIP))) {
30             arr[j+1] = arr[j];
31             j--;
32         }
33         arr[j+1] = key;
34     }
35 }
```

Output

```
Daftar Pegawai Sebelum Terurut:
NIP: 23, Nama: Rayhan Elmo, Gender: L
NIP: 1, Nama: Oscar Javier, Gender: L
NIP: 6, Nama: Fina Salsa, Gender: P
NIP: 11, Nama: Tora Kamulian, Gender: L
NIP: 4, Nama: Hatfan Sahrul, Gender: L
NIP: 9, Nama: Tiyyasha Ananda, Gender: P
-----
Pilih Metode Pengurutan:
a. Berdasarkan NIP
b. Berdasarkan Nama
Masukkan pilihan: a
Pilih Urutan:
a. Naik
b. Turun
Masukkan pilihan: a

Daftar Pegawai Setelah Terurut:
NIP: 1, Nama: Oscar Javier, Gender: L
NIP: 4, Nama: Hatfan Sahrul, Gender: L
NIP: 6, Nama: Fina Salsa, Gender: P
NIP: 9, Nama: Tiyyasha Ananda, Gender: P
NIP: 11, Nama: Tora Kamulian, Gender: L
NIP: 23, Nama: Rayhan Elmo, Gender: L
-----
Code Execution Successful
```

Output

Kesimpulan

Eksplorasi program-program sebelumnya memberikan gambaran yang kaya tentang penggunaan berbagai metode pengurutan dalam bahasa pemrograman C++. Melalui struktur data Pegawai dan vektor standar, kita dapat dengan fleksibel menyimpan dan mengelola data. Meskipun Selection Sort menawarkan fleksibilitas dalam pengurutan berdasarkan kriteria yang berbeda, efisiensi algoritma tetap menjadi pertimbangan penting, terutama untuk data yang besar.

Interaksi yang ditingkatkan dengan pengguna, menggunakan loop do-while, memudahkan mereka untuk memilih metode pengurutan tanpa keluar dari program. Praktik pemrograman yang baik, seperti penanganan input yang efektif dan organisasi logis menggunakan fungsi bantu, juga diterapkan untuk meningkatkan kejelasan dan pemeliharaan kode.

Keseluruhan inisiatif ini memberikan pemahaman yang lebih dalam tentang pengurutan dan manipulasi data, serta mengilustrasikan aplikasi konsep-konsep tersebut dalam situasi dunia nyata. Ini memberikan dasar yang kuat untuk pengembangan perangkat lunak dan pemecahan masalah dengan menggunakan C++.