Praktikum Algoritma Struktur Data

Praktikum 07:Bubble dan Shell Sort

Oleh:

Bintang Bimantara/ 5223600025

Program Studi Sarjana Terapan Teknologi Game

Departemen Teknologi Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2024

Percobaan

1. Percobaan 1 : Implementasi pengurutan dengan metode *Bubble Sort.*

main.cpp

Run

```cpp
#include <iostream>
#include <cstdlib>
#define MAX 10
using namespace std;
//Nama: Bintang Bimantara
//NRP:52235600025

int Data[MAX];

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

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

```
27  }
28
29  int main()
30  {
31      int i;
32      srand(0);
33      // Membangkitkan bilangan acak
34      cout << "DATA SEBELUM TERURUT" << endl;
35      for (i = 0; i < MAX; i++)
36      {
37          Data[i] = rand() / 1000 + 1;
38          cout << "Data ke " << i << " : " << Data[i] << endl;
39      }
40      BubbleSort();
41      // Data setelah terurut
42      cout << "DATA SETELAH TERURUT" << endl;
43      for (i = 0; i < MAX; i++)
44      {
45          cout << "Data ke " << i << " : " << Data[i] << endl;
46      }
47      return 0;
48  }
49
```

Hasil Output:

```
Output
/tmp/EVFrgEcOFs.o
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 ===
```

2. Percobaan 2 : Implementasi pengurutan dengan metode penyisipan Shell Sort.

```cpp
#include <iostream>
#include <cstdlib>
#define MAX 10
using namespace std;
//Nama: Bintang Bimantara
//NRP: 5223600025
int Data[MAX];

// 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()
{
    int i;
    srand(0);
    // Membangkitkan bilangan acak
    cout << "DATA SEBELUM TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        Data[i] = rand() / 1000 + 1;
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    ShellSort();
    // Data setelah terurut
    cout << "DATA SETELAH TERURUT" << endl;
    for (i = 0; i < MAX; i++)
    {
        cout << "Data ke " << i << " : " << Data[i] << endl;
    }
    return 0;
}

```

Hasil Output:

```
Output
/tmp/1mdSOqBsfQ.o
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. Tambahkan kode program untuk menampilkan perubahan setiap iterasi dari proses pengurutan dengan Bubble Sort dan Shell Sort.

```
23
24          // Menampilkan larik setiap kali terjadi pertukaran
25          cout << "Iterasi " << i << ": ";
26          for(int k = 0; k < MAX; k++) {
27              cout << Data[k] << " ";
28          }
29          cout << endl;
```

   untuk menampilkan perubahan setiap iterasi dari proses pengurutan dalam penyisipan akan diberi kode ini yang akan menampilkan setiap iterasi yang terjadi.

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

```
int Data[MAX];
int comparisonCount = 0; // Variabel untuk menghitung banyaknya perbandingan
int shiftCount = 0; // Variabel untuk menghitung banyaknya pergeseran

```

untuk menghitung berapa banyak perbandingan dan pergeseran maka pertama perlu mendeklarasi sebuah variabel untuk menghitung tersebut.

```
        Sudah = true;
        for (j = 0; j < MAX - Jarak; j++)
        {
            shiftCount++; // Menambah jumlah pergeseran
            i = j + Jarak;
            if (Data[j] > Data[i])
            {
                Tukar(&Data[j], &Data[i]);
                Sudah = false;
                comparisonCount++; // Menambah jumlah perbandingan

                // Menampilkan larik setiap kali terjadi pertukaran
                cout << "Iterasi: "<< i << ": ";
                for(int k = 0; k < MAX; k++) {
                    cout << Data[k] << " ";
                }
                cout << endl;
            }
        }
    }
}
```

Kemudian menambahkan isi dari variabel tersebut pada fungsi yang melakukan pengurutan.

```
56
57      // Menampilkan jumlah perbandingan dan pergeseran
58      cout << "\nJumlah Perbandingan: " << comparisonCount << endl;
59      cout << "Jumlah Pergeseran: " << shiftCount << endl;
60      return 0;
61  }
```

Setelah itu tampilkan hasil variabel tersebut dengan cout pada int main.

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.

main.cpp

Run

```cpp
#include <iostream>
#include <string>
#include <algorithm>
//Nama:Bintang Bimantara
//NRP: 5223600025
using namespace std;

// Definisi struktur Pegawai
struct Pegawai {
    string NIP;
    string nama;
};

// Fungsi untuk menampilkan daftar pegawai
void tampilkanDaftar(const Pegawai daftar[], int size) {
    for (int i = 0; i < size; ++i) {
        cout << "NIP: " << daftar[i].NIP << ", Nama: " << daftar[i].nama << endl;
    }
}

// Fungsi untuk membandingkan pegawai berdasarkan NIP atau nama
bool bandingkanPegawai(const Pegawai& a, const Pegawai& b, bool berdasarkanNIP, bool urutNaik) {
    if (berdasarkanNIP) {
        if (urutNaik) {
            return a.NIP < b.NIP;
        } else {
            return a.NIP > b.NIP;
```

```
        }
    } else {
        if (urutNaik) {
            return a.nama < b.nama;
        } else {
            return a.nama > b.nama;
        }
    }
}

// Prosedur pengurutan metode gelembung
void BubbleSort(Pegawai daftar[], int size, bool berdasarkanNIP, bool urutNaik)
{
    for (int i = 0; i < size - 1; ++i) {
        for (int j = 0; j < size - i - 1; ++j) {
            if (bandingkanPegawai(daftar[j], daftar[j + 1], berdasarkanNIP, urutNaik))
                {
                swap(daftar[j], daftar[j + 1]);
            }
        }
    }
}

// Prosedur pengurutan metode Shell
void ShellSort(Pegawai daftar[], int size, bool berdasarkanNIP, bool urutNaik)
{
    for (int gap = size / 2; gap > 0; gap /= 2) {
```

```
        for (int i = gap; i < size; ++i) {
            Pegawai temp = daftar[i];
            int j;
            for (j = i; j >= gap && bandingkanPegawai(daftar[j - gap], temp,
                berdasarkanNIP, urutNaik); j -= gap) {
                daftar[j] = daftar[j - gap];
            }
            daftar[j] = temp;
        }
    }
}
```

```cpp

int main() {
    const int SIZE = 5;
    Pegawai data[SIZE] = {
        {"123", "Joni"},
        {"456", "Siti"},
        {"789", "Sugeng"},
        {"234", "Fara"},
        {"567", "Steven"}
    };

    int pilihan;
    cout << "Pilih metode pengurutan:" << endl;
    cout << "1. Bubble Sort" << endl;
    cout << "2. Shell Sort" << endl;
    cin >> pilihan;

    bool berdasarkanNIP;
    cout << "Pilih berdasarkan NIP? (1: Ya, 0: Tidak): ";
    cin >> berdasarkanNIP;

    bool urutNaik;
    cout << "Pilih urutan: (1: Naik, 0: Turun): ";
    cin >> urutNaik;

```

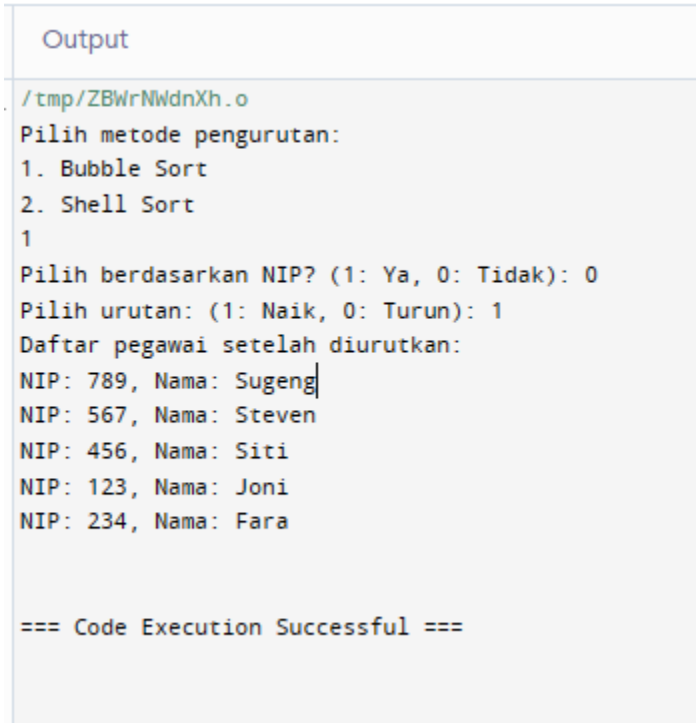
```cpp
    switch (pilihan) {
        case 1:
            BubbleSort(data, SIZE, berdasarkanNIP, urutNaik);
            break;
        case 2:
            ShellSort(data, SIZE, berdasarkanNIP, urutNaik);
            break;
        default:
            cout << "Pilihan tidak valid!" << endl;
            return 1;
    }

    // Menampilkan daftar pegawai yang sudah diurutkan
    cout << "Daftar pegawai setelah diurutkan:" << endl;
    tampilkanDaftar(data, SIZE);

    return 0;
}

```

Hasil Output:



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

   Bubble Sort adalah algoritma pengurutan yang sederhana tetapi kurang efisien, karena melakukan pertukaran secara bertahap hingga seluruh daftar terurut. Shell Sort, di sisi lain, menggunakan strategi yang lebih cerdas dengan mengurutkan sub daftar terpisah menggunakan jarak tertentu, yang menghasilkan kinerja yang lebih baik, meskipun implementasinya sedikit lebih kompleks. Jadi, jika kita memiliki data yang tidak terlalu besar, kita bisa

menggunakan Bubble Sort, tetapi jika kita membutuhkan kinerja yang lebih cepat, Shell Sort adalah pilihan yang lebih baik.