SHELL SORT

Nama : Reiznu Ahmad Tjandrida
NIM : 21091397018 - **2021 B**

Source Code Shell Sort

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

// Shell Sort
void ShellSort(int data[], int jumlah_data)
{
    // Menentukan variabel index dengan index array yang akan dibandingkan nilainya
    int index, perulangan_kedua, perulangan_ketiga, angka_sementara;

    for(index = jumlah_data/2; index > 0; index = index/2)
    {
        // Perulangan Kedua
        for(perulangan_kedua = index; perulangan_kedua < jumlah_data; perulangan_kedua++)
        {
            // Perulangan Ketiga
            for(perulangan_ketiga = perulangan_kedua-index; perulangan_ketiga >= 0; perulangan_ketiga = perulangan_ketiga-index)
            {
                // Jika nilai dari data[perulangan_ketiga+index] lebih besar dari data[perulangan_ketiga]
                if(data[perulangan_ketiga+index] >= data[perulangan_ketiga])
                break;
                // Menukar nilai
                else
                {
                    angka_sementara = data[perulangan_ketiga];
                    data[perulangan_ketiga] = data[perulangan_ketiga+index];
                    data[perulangan_ketiga+index] = angka_sementara;
                }
            }
        }
    }
}


int main()
{

    int jumlah_data, index;
    cout<<"\n Masukkan Jumlah Data: ";
    cin>>jumlah_data;

    int data[jumlah_data];
    for(index = 0; index < jumlah_data; index++)
    {
        cout<<"Masukkan Elemen Ke "<<index+1<<": ";
        cin>>data[index];
    }

    ShellSort(data, jumlah_data);

    // Menampilkan data yang telah di urutkan
    cout<<"\n Data Yang Telah Di Urutkan ";
    for (index = 0; index < jumlah_data; index++)
        cout << " " << data[index];

    return 0;
}
```

PENJELASAN KODE

Pada baris ke 5 sampai baris 31 terdapat sebuah perintah yang berfungsi untuk melakukan logika pengurutan ***Shell Sort.***

```
4   // Shell Sort
5   void ShellSort(int data[], int jumlah_data)
6   {
7       // Menentukan variabel index dengan index array yang akan dibandingkan nilainya
8       int index, perulangan_kedua, perulangan_ketiga, angka_sementara;
9
10      for(index = jumlah_data/2; index > 0; index = index/2)
11      {
12          // Perulangan Kedua
13          for(perulangan_kedua = index; perulangan_kedua < jumlah_data; perulangan_kedua++)
14          {
15              // Perulangan Ketiga
16              for(perulangan_ketiga = perulangan_kedua-index; perulangan_ketiga >= 0; perulangan_ketiga = perulangan_ketiga-index)
17              {
18                  // Jika nilai dari data[perulangan_ketiga+index] lebih besar dari data[perulangan_ketiga]
19                  if(data[perulangan_ketiga+index] >= data[perulangan_ketiga])
20                  break;
21                  // Menukar nilai
22                  else
23                  {
24                      angka_sementara = data[perulangan_ketiga];
25                      data[perulangan_ketiga] = data[perulangan_ketiga+index];
26                      data[perulangan_ketiga+index] = angka_sementara;
27                  }
28              }
29          }
30      }
31  }
32
```

A. Pada baris ke 8 terdapat sebuah kode yang berisi variabel **index, perulangan_kedua, perulangan_ketiga, angka_sementara**

Pada baris ke 10 terdapat variabel **index** berfungsi untuk menentukan jumlah index angka sesuai variabel **jumlah_data.**

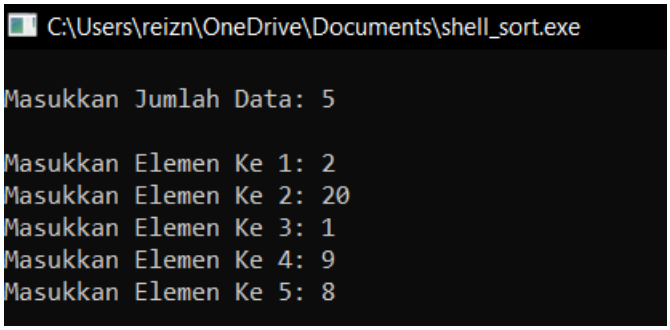
```
7       // Menentukan variabel index dengan index array yang akan dibandingkan nilainya
8       int index, perulangan_kedua, perulangan_ketiga, angka_sementara;
9
```

B. Pada baris ke 13 sampai baris 29 terdapat sebuah perintah untuk melakukan perulangan terhadap angka yang di inputkan oleh user yang kemudian di sorting sesuai perintah yang kita tulis nanti.

```
12          // Perulangan Kedua
13          for(perulangan_kedua = index; perulangan_kedua < jumlah_data; perulangan_kedua++)
14          {
15              // Perulangan Ketiga
16              for(perulangan_ketiga = perulangan_kedua-index; perulangan_ketiga >= 0; perulangan_ketiga = perulangan_ketiga-index)
17              {
18                  // Jika nilai dari data[perulangan_ketiga+index] lebih besar dari data[perulangan_ketiga]
19                  if(data[perulangan_ketiga+index] >= data[perulangan_ketiga])
20                  break;
21                  // Menukar nilai
22                  else
23                  {
24                      angka_sementara = data[perulangan_ketiga];
25                      data[perulangan_ketiga] = data[perulangan_ketiga+index];
26                      data[perulangan_ketiga+index] = angka_sementara;
27                  }
28              }
29          }
30      }
31  }
```

- Pada baris ke 13, terdapat variabel **perulangan_kedua** yang berisikan variabel **index** dari baris ke 8. Jadi, nantinya

variabel **index** akan ditampung oleh variabel **perulangan_kedua.** Kemudian looping dari **perulangan_ kedua** tidak akan lebih dari variabel **jumlah_data.**
CONTOH : jika isi dari variabel **jumlah_data** = 5, maka looping untuk variabel **perulangan_kedua** tidak melebihi dari angka 5.



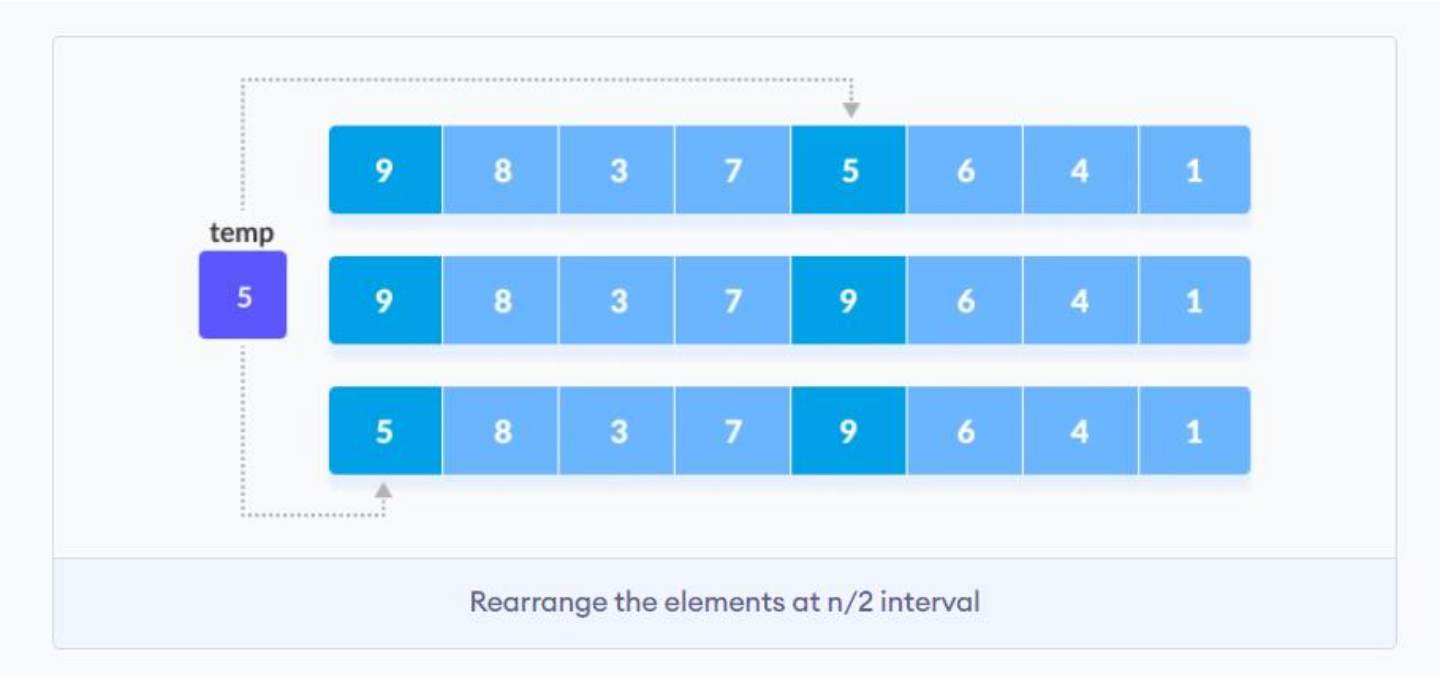
Jadi, variabel **jumlah_data** berfungsi untuk menampung inputan angka dari user yang kemudian dijadikan jumlah maksimal looping untuk baris ke 13 sampai 29.

C. Pada baris 34 sampai 57, terdapat fungsi utama pada program ini.
-Baris 37 terdapat variabel **jumlah_data** dan **index**
-Baris 38 terdapat cout yang berfungsi untuk mencetak tulisan “Masukkan Jumlah Data : “
-Baris 39 terdapat perintah yang berfungsi untuk menampung inputan dari user yang kemudian ditampung di variabel **jumlah_data.**
-Baris 42 terdapat variabel **data** yang nantinya berisikan array nilai dari variabel **jumlah_data**
-Baris 43 terdapat perintah untuk melakukan perulangan untuk setiap inputan elemen array
-Baris 49 terdapat perintah untuk memanggil function **ShellSort** dengan mengrimkan sebuah parameter **data** dan **jumlah_data**
-Baris 53 sampai 56 terdapat perulangan for yang berfungsi untuk menampilkan data yang sudah di urutkan
-Kemudian pada baris 58 terdapat perintah untuk mengembalikan sebuah nilai

```
34  int main()
35  {
36
37      int jumlah_data, index;
38      cout<<"\nMasukkan Jumlah Data: ";
39      cin>>jumlah_data;
40      cout << endl;
41
42      int data[jumlah_data];
43      for(index = 0; index < jumlah_data; index++)
44      {
45          cout<<"Masukkan Elemen Ke "<<index+1<<": ";
46          cin>>data[index];
47      }
48
49      ShellSort(data, jumlah_data);
50
51      // Menampilkan data yang telah di urutkan
52      cout<<"\nData Yang Telah Di Urutkan ";
53      for (index = 0; index < jumlah_data; index++)
54      {
55          cout << " " << data[index];
56      }
57
58      return 0;
59  }
```

Data Yang Belum Diurutkan

Proses menukar data sesuai interval yang telah di tentukan



Proses ini berlangsung untuk semua elemen data

Proses Pengurutan Shell Sort

Algoritam Shell Sort

```
shellSort(array, size)
    for interval i <- size/2n down to 1
    for each interval "i" in array
        sort all the elements at interval "i"
end shellSort
```

Shell Sort Time Complexity

Time Complexity	
Best	O(nlog n)
Worst	$O(n^2)$
Average	O(nlog n)
Space Complexity	O(1)
Stability	No

BIG O – Worst Case

A. N = 1

n = 1
log(1) = 0
1 = 1
1 log(1) = 0
1^2 = 1
2^1 = 2
1! = 1

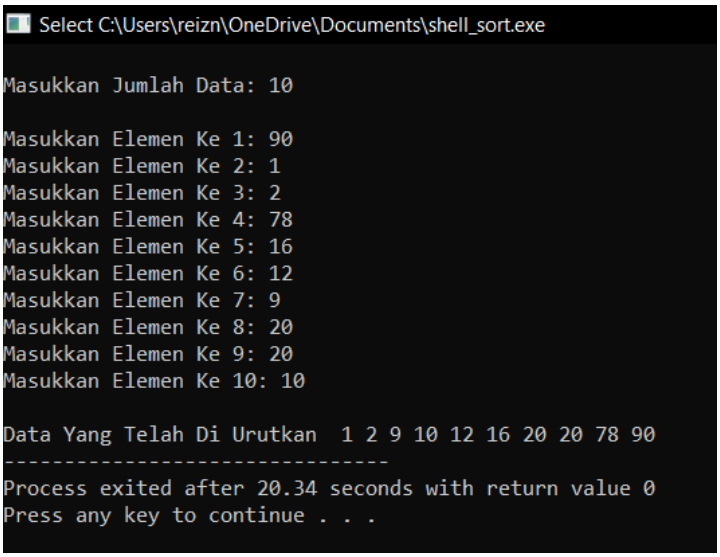
B. N = 5

n = 5
log(5) = 0,698
5 = 5
5 log(5) = 3,49
5^2 = 25
2^5 = 32
5! = 120

C. N = 10

n = 10
log(10) = 1
10 = 10
10 log(10) = 10
10^2 = 100
2^10 = 1024
10! = 3628800

Hasil Kodingan



Kelebihan Dan Kekurangan

KELEBIHAN :

1. Algoritma ini sangat rapat dan mudah untuk diimplementasikan.

2. Operasi pertukarannya hanya dilakukan sekali saja.

3. Waktu pengurutan dapat lebih ditekan.

4. Mudah menggabungkannya kembali.

5. Kompleksitas selection sort relatif lebih kecil.

KEKURANGAN :

1. Membutuhkan method tambahan.

2. Sulit untuk membagi masalah.