

Nama : Au Izaldi Fachril Rahmadani

Kelas : B

NIM : 21091397026

Prodi : D4 Manajemen Informatika

Shell Sort

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

void ShellSort(int sort[], int size){ //fungsi shell sort dengan parameter sort[] dan size masing-masing
bertipe data INT
    for(int gap = size/2; gap > 0; gap /= 2){
        for(int x = gap; x < size; x++){
            int temp = sort[x];
            int y;
            for(y = x; y >= gap && sort[y-gap] > temp; y -= gap){
                sort[y] = sort[y-gap];
            }
            sort[y] = temp;
        }
    }
}

//main program
int main(){
    //untuk menampilkan batas array
    int ukuran;
    cout<<"Masukkan Batas Array\t: ";
    cin>>ukuran;

    //deklarasi variabel array (sort) dengan batas elemen dari inputan variabel (size)
    int sort[ukuran];
    cout<<"Masukkan Elemen Array\t: ";
    /*
    perulangan for inisialisasi x dimulai 0 dengan batas x kurang dari ukuran
    dan x diincrement
    */
    for(int x = 0; x < ukuran; x++){
        cin>>sort[x];
    }
    ShellSort(sort, ukuran);//memanggil fungsi ShellSort
    cout<<"Array Setelah diSortir\t: ";
    for(int x = 0; x < ukuran; x++){
        cout<<"["<<sort[x]<<" "<<"\t";
    }
}
```

Output

```
C:\Users\Dani PC\Documents\C++\shell.exe
Masukkan Batas Array : 6
Masukkan Elemen Array : 45 12 67 22 11 10
Array Setelah diSortir : [10] [11] [12] [22] [45] [67]
-----
Process exited after 44.77 seconds with return value 0
Press any key to continue . . .
```

Shell sort merupakan salah satu algoritma pengurutan data dengan cara membandingkan suatu data dengan data lainnya yang memiliki jarak tertentu dan pada langkah selanjutnya elemen yang dibandingkan akan semakin kecil, sehingga jarak antar elemen yang dibandingkan sudah sama dengan 1.

Cara kerja

| Jarak | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 |
|-----------------|--------|--------|--------|--------|--------|--------|
| Awal | 6 | 12 | 67 | 22 | 11 | 10 |
| Jarak $6/2 = 3$ | 6 | 12 | 67 | 22 | 11 | 10 |
| | 6 | 12 | 67 | 22 | 11 | 10 |
| | 6 | 11 | 67 | 22 | 12 | 10 |
| | 6 | 11 | 10 | 22 | 12 | 67 |
| Jarak $3/2 = 1$ | 6 | 11 | 10 | 22 | 12 | 67 |
| | 6 | 11 | 10 | 22 | 12 | 67 |
| | 6 | 10 | 11 | 22 | 12 | 67 |
| | 6 | 10 | 11 | 22 | 12 | 67 |
| | 6 | 10 | 11 | 12 | 22 | 67 |
| Akhir | 6 | 10 | 11 | 12 | 22 | 67 |

1. Awal perbandingan jarak 3 diperoleh dari jumlah elemen dibagi dengan 2, data 1 = 6 dibandingkan dengan data 4 = 22 jika nilai dari data 1 lebih kecil dari data 4 maka akan ditukar, jika tidak maka dilanjutkan ke perbandingan selanjutnya.
2. Bandingkan data 2 = 12 dan data 5 = 11 jika nilai dari data 2 lebih kecil dari data 5 maka akan ditukar, jika tidak maka dilanjutkan ke perbandingan selanjutnya.

3. Bandingkan data 3 = 67 dan data 6 = 10 jika nilai dari data 3 lebih kecil dari data 6 maka akan ditukar, jika tidak maka dilanjut ke perbandingan selanjutnya.
4. Dan jika semua sudah berurutan sesuai dengan yang dibandingkan dengan jarak tiga maka lanjut dengan jarak 1.
5. Seperti sebelumnya tetapi sekarang dibandingkan dengan data yang di sebelahnya seperti data 1 dan data 2 jika sudah terurut, maka dilanjut data 3 dan data 4, dan seterusnya sampai data sudah terurut.

Penjelasan program

1. Terdapat fungsi *void ShellSort* yang digunakan sebagai pengurut array didalamnya terdapat *nested loop*
2. Untuk *main programnya* terdapat program untuk memasukkan batas array yang akan disort.
3. Kemudian terdapat program untuk menginputkan elemen-elemen array sesuai dengan batasnya yang sudah diinputkan tadi
4. Setelah itu ada pemanggilan fungsi *ShellSort* tadi kemudian ada keluaran yang berupa array yang sudah disortir.

Kelebihan

1. Algoritma sangat mudah diimplementasikan
2. Waktu pengurutan dapat lebih ditekan
3. Algoritmanya lebih sederhana dibandingkan merge sort

Kekurangan

1. Membutuhkan method tambahan
2. Sulit untuk membagi masalah