Laporan Individu Struktur Data

(Radix Sort)

Nama : Muhammad Yuda Pratama
NIM : 21091397025
Kelas : A

Code Radix Sort :

Radix sort.cpp

```cpp
//Muhammad Yuda Pratama
//21091397025
//A

#include <iostream>
using namespace std;

//fungsi untuk mendapatkan nilai maksimum dari array
int getMax(int arr[], int size)
{
    int max = arr[0];
    for (int i = 1; i < size; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}
```

```cpp
//fungsi untuk melakukan perhitungan array sesuai digit
void CountingSort(int arr[], int size, int div)
{
    int output[size];
    int count[10] = {0};

    //simpan jumlah kemunculan dalam hitungan
    for (int i = 0; i < size; i++)
        count[ (arr[i]/div)%10 ]++;

    //mengubah hitungan sehingga hitungan tersebut berisi dan
    //posisi hitungan menjadi output
    for (int i = 1; i < 10; i++)
        count[i] += count[i - 1];

    //membangun output array
    for (int i = size - 1; i >= 0; i--)
    {
        output[count[ (arr[i]/div)%10 ] - 1] = arr[i];
        count[ (arr[i]/div)%10 ]--;
    }

    //menyalin output array ke arr[i] sehingga arr[i] berisi nomor yang diurutkan menurut digit
    for (int i = 0; i < size; i++)
        arr[i] = output[i];
}
```

```
45  void RadixSort(int arr[], int size)
46  {
47  //memanggil pengurutan perhitungan beberapa kali dan berapa kali perhitungan disebut
48  //akan sama dengan jumlah digit yang dimiliki jumlah maksimum dalam array
49      int m = getMax(arr, size);
50      for (int div = 1; m/div > 0; div *= 10)
51          CountingSort(arr, size, div);       //menghitung sort
52  }
```

```
54  int main()
55  {
56      cout << "======= Radix Sort ========" << endl;
57      cout << endl;
58
59      int size;       //inisialisasi variable
60      cout << "Masukkan Jumlah Array : " << endl;
61      cin >> size;    //input jumlah array
62      int arr[size];
63
64      cout << "Masukkan Nilai Array : " << endl;
65      //looping nilai array yang dimasukkan user
66      for(int i=0; i<size; i++){
67          cin >> arr[i];
68      }
69      cout << endl;
70
71      //memanggil function radix sort
72      RadixSort (arr, size);
73      cout << "Hasil array yang sudah disorting adalah : " << endl;
74
75      //looping hasil yang telah disorting
76      for(int i=0; i<size; i++){
77          cout << "[" << arr[i] << "]" << " ";
78      }
79      return 0;
80  }
```

Output Program :

```
======= Radix Sort ========

Masukkan Jumlah Array : 5
Masukkan Nilai Array :
38 15 27 10 8

Hasil array yang sudah disorting adalah :
[8] [10] [15] [27] [38]
--------------------------------
Process exited after 29.21 seconds with return value 0
Press any key to continue . . .
```

Kelebihan Radix Sort

1. Algoritma sangat praktis. Hal ini dapat dilihat dari kompleksitas waktu asimptotiknya yang sangat kecil (O(kN)). Hal ini mengakibatkan algoritma radix sort sangat efektif untuk data dalam jumlah yang sangat besar sekalipun.
2. Konsep algoritma mudah dipahami. Algoritma radix sort mengurutkan data berdasarkan digit, tidak melalui proses perbandingan yang cenderung sulit dipahami.

Kekurangan Radix Sort

1. Realisasi program rumit dan kurang fleksibel untuk digunakan pada tipe data lain. Algoritma radix sort untuk tipe data lain harus dibantu dengan menggunakan counting sort, yang merupakan salah satu algoritma pengurutan yang tidak menggunakan perbandingan.
2. Realisasi program untuk algoritma radix sort tidak semudah memahami konsep dasarnya.
3. Algoritma ini membutuhkan bucket untuk mengelompokkan data-data yang sedang diurutkan. Inisialisasi bucket untuk kasus ini tidak mudah dilakukan.