# Praktikum 7

# Pengurutan (Sorting)

# Insertion Sort, Selection Sort

## POKOK BAHASAN:

- ✓ Konsep pengurutan dengan *insertion sort* dan *selection sort*
- ✓ Struktur data proses pengurutan
- ✓ Implementasi algoritma pengurutan *insertion sort* dan *selection sort* dalam Bahasa C

## TUJUAN BELAJAR:

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami karakteristik algoritma pengurutan *straight insertion sort, binary insertion sort* dan *selection sort.*
- ✓ Mengimplementasikan algoritma *straight insertion sort, binary insertion sort* dan *selection sort* dalam bentuk flowchart.
- ✓ Membuat diagram alir dan mengimplementasikan algoritma pada suatu permasalahan.

## DASAR TEORI:

Pengurutan data (*sorting*) didefinisikan sebagai suatu proses untuk menyusun kembali humpunan obyek menggunakan aturan tertentu.

Ada dua macam urutan yang biasa digunakan dalam proses pengurutan yaitu

- urut naik (*ascending*) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar
- urut turun (*descending*) yaitu data yang mempunyai nilai paling besar sampai paling kecil.

Deklarasi larik yang digunakan adalah larik dimensi satu (*vektor*) dengan elemennya bertipe integer.

```
#define  MAX 100;
int Data[MAX];
```

Pada deklarasi diatas, `MAX` adalah banyaknya elemen vektor. Anda bisa mengubah nilai konstanta `MAX` sesuai kebutuhan. Indeks larik dimulai dari 0. Data yang sebenarnya disimpan mulai dari indeks 0.

Selain deklarasi diatas, proses yang juga selalu digunakan pada algoritma pengurutan adalah proses menukarkan elemen. Di bawah ini satu prosedur sederhana untuk menukarkan nilai dua buah elemen A dan B.

```
void Tukar (int *a, int *b)
{
      int temp;
      temp = *a;
      *a = *b;
      *b = temp;
}
```

## 1. METODE PENYISIPAN (*INSERTION SORT)*

Terdapat 2 macam metode penyisipan, yaitu metode penyisipan langsung (*straight insertion sort*) dan metode penyisipan biner (*binary insertion sort*).

### 1.1 LANGSUNG (*STRAIGHT INSERTION SORT*)

Ilustrasi dari langkah-langkah pengurutan dengan algoritma penyisipan langsung (*straight insertion sort*) dapat dilihat pada tabel berikut :

| Iterasi | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] | Data[8] | Data[9] |
|---|---|---|---|---|---|---|---|---|---|---|
| Awal | 12 | 35 | 9 | 11 | 3 | 17 | 23 | 15 | 31 | 20 |
| i=1 | 12 | **35** | 9 | 11 | 3 | 17 | 23 | 15 | 31 | 20 |
| i=2 | 12 | 35 | **9** | 11 | 3 | 17 | 23 | 15 | 31 | 20 |
| i=3 | 9 | 12 | 35 | **11** | 3 | 17 | 23 | 15 | 31 | 20 |
| i=4 | 9 | 11 | 12 | 35 | **3** | 17 | 23 | 15 | 31 | 20 |
| i=5 | 3 | 9 | 11 | 12 | 35 | **17** | 23 | 15 | 31 | 20 |
| i=6 | 3 | 9 | 11 | 12 | 17 | 35 | **23** | 15 | 31 | 20 |
| i=7 | 3 | 9 | 11 | 12 | 17 | 23 | 35 | **15** | 31 | 20 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| i=8 | 3 | 9 | 11 | 12 | 15 | 17 | 23 | 35 | **31** | 20 |
| i=9 | 3 | 9 | 11 | 12 | 15 | 17 | 23 | 31 | 35 | **20** |
| Akhir | 3 | 9 | 11 | 12 | 15 | 17 | 20 | 23 | 31 | 35 |

Data pada posisi ke *i* (x) dibandingkan dengan data pada posisi ke 0 sampai dengan *i*–1. Apabila data ke *j* lebih besar daripada x, maka data disisipkan ke posisi ke *j* dan data ke *j*+1 sampai dengan *i* digeser ke kanan.

## 1.2 METODE PENYISIPAN BINER (*BINARY INSERTION SORT*)

Metode pengurutan dengan algoritma penyisipan biner (*binary insertion sort*) memperbaiki metode pengurutan dengan algoritma penyisipan langsung dengan melakukan proses perbandingan yang lebih sedikit sehingga proses pengurutan lebih cepat.

Metode penyisipan biner melakukan proses perbandingan dengan membagi dua bagian data dari posisi 0 sampai dengan *i*-1 yang disebut dengan bagian kiri dan kanan. Apabila data pada posisi ke *i* berada pada jangkauan kiri maka proses perbandingan dilakukan hanya pada bagian kiri dan menggeser posisi sampai *i*.

## 2. METODE SELEKSI (*SELECTION SORT*)

Ilustrasi langkah-langkah pengurutan dengan algoritma seleksi (*selection sort*) dapat dilihat pada tabel berikut :

| Iterasi | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] | Data[8] | Data[9] |
|---|---|---|---|---|---|---|---|---|---|---|
| Awal | 12 | 35 | 9 | 11 | 3 | 17 | 23 | 15 | 31 | 20 |
| I=0 | **12** | 35 | 9 | 11 | **3** | 17 | 23 | 15 | 31 | 20 |
| I=1 | 3 | **35** | **9** | 11 | 12 | 17 | 23 | 15 | 31 | 20 |
| I=2 | 3 | 9 | **35** | **11** | 12 | 17 | 23 | 15 | 31 | 20 |
| I=3 | 3 | 9 | 11 | **35** | **12** | 17 | 23 | 15 | 31 | 20 |
| I=4 | 3 | 9 | 11 | 12 | **35** | 17 | 23 | **15** | 31 | 20 |
| I=5 | 3 | 9 | 11 | 12 | 15 | **17** | 23 | 35 | 31 | 20 |
| I=6 | 3 | 9 | 11 | 12 | 15 | 17 | **23** | 35 | 31 | **20** |
| I=7 | 3 | 9 | 11 | 12 | 15 | 17 | 20 | **35** | 31 | **23** |
| I=8 | 3 | 9 | 11 | 12 | 15 | 17 | 20 | 23 | **31** | 35 |
| Akhir | 3 | 9 | 11 | 12 | 15 | 17 | 20 | 23 | 31 | 35 |

Mula-mula ditentukan posisi ke i. Kemudian dilakukan penelusuran data terkecil dan dilakukan pertukaran data. Posisi *i* dinaikkan dan proses penelusuran dan pertukaran dilakukan sampai posisi terakhir.

## TUGAS PENDAHULUAN:

1. Buatlah flowchart untuk operasi pengurutan bilangan bulat dengan algoritma *straight insertion sort, binary insertion sort* dan *selection sort*.
2. Buatlah deklarasi data pegawai dan flowchart fungsi untuk mengurutkan data dengan metode *insertion sort (straight insertion dan binary insertion sort)* dan *selection sort* sebagai berikut :
   - Data bertipe rekaman bernama Pegawai yang mempunyai 4 data yaitu :
     - `NIP`, bertipe bulat
     - `Nama`, bertipe string
     - `Alamat`, bertipe string
     - `Golongan`, bertipe char
   - Data diurutkan dengan urut naik berdasarkan NIP

## PERCOBAAN:

1. Buatlah workspace menggunakan Visual C++.
2. Buatlah project untuk praktikum SORTING dan file *C Source* untuk metode pengurutan *straight insertion sort, binary insertion sort* dan *selection sort*.
3. Cobalah untuk masing-masing percobaan di bawah dengan menambahkan menu pilihan metode pengurutan pada program utama.

### Percobaan 1 : Implementasi pengurutan dengan metode penyisipan langsung (*straight insertion sort*)

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

int Data[MAX];
```

```
// Fungsi pengurutan penyisipan langsung

void StraightInsertSort()
{
      int i, j, x;
      for(i=1; i<MAX; i++){
            x = Data[i];
            j = i - 1;
            while (x < Data[j]){
                  Data[j+1] = Data[j];
                  j--;
            }
            Data[j+1] = x;
      }
}

void main()
{
   int i;
   srand(0);

   // Membangkitkan bilangan acak
   printf("DATA SEBELUM TERURUT");
   for(i=0; i<MAX; i++)
   {
         Data[i] = (int) rand()/1000+1;
         printf("\nData ke %d : %d ", i, Data[i]);
   }

   StraightInsertSort();

   // Data setelah terurut
   printf("\nDATA SETELAH TERURUT");
   for(i=0; i<MAX; i++)
   {
         printf("\nData ke %d : %d ", i, Data[i]);
   }
}
```

Percobaan 2 : Implementasi pengurutan dengan metode penyisipan biner (*binary insertion sort*)

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

int Data[MAX];
```

```
// Fungsi pengurutan penyisipan biner
void BinaryInsertSort()
{
      int i, j, l, r, m, x;
      for (i=1; i<MAX; i++){
            x = Data[i];
            l = 0;
            r = i - 1;
            while(l <= r){
                  m = (l + r) / 2;
                  if(x < Data[m])
                        r = m - 1;
                  else
                        l = m + 1;
            }
            for(j=i-1; j>=l; j--)
                  Data[j+1] = Data[j];
            Data[l]=x;
      }
}

void main()
{
   int i;
   srand(0);

   // Membangkitkan bilangan acak
   printf("DATA SEBELUM TERURUT");
   for(i=0; i<MAX; i++)
   {
         Data[i] = (int) rand()/1000+1;
         printf("\nData ke %d : %d ", i, Data[i]);
   }

   BinaryInsertSort();

   // Data setelah terurut
   printf("\nDATA SETELAH TERURUT");
   for(i=0; i<MAX; i++)
   {
         printf("\nData ke %d : %d ", i, Data[i]);
   }
}
```

Percobaan 3 : Implementasi pengurutan dengan metode seleksi (*selection sort*)

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

int Data[MAX];
```

```
// Fungsi pertukaran bilangan

void Tukar (int *a, int *b)
{
      int temp;
      temp = *a;
      *a = *b;
      *b = temp;
}

// Fungsi pengurutan penyisipan biner

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

void main()
{
   int i;
   srand(0);

   // Membangkitkan bilangan acak
   printf("DATA SEBELUM TERURUT");
   for(i=0; i<MAX; i++)
   {
         Data[i] = (int) rand()/1000+1;
         printf("\nData ke %d : %d ", i, Data[i]);
   }

   SelectionSort();

   // Data setelah terurut
   printf("\nDATA SETELAH TERURUT");
   for(i=0; i<MAX; i++)
   {
         printf("\nData ke %d : %d ", i, Data[i]);
   }
}
```

## LATIHAN:

1. Tambahkan kode program untuk menampilkan perubahan setiap iterasi dari proses pengurutan dengan penyisipan langsung, penyisipan biner dan seleksi.
2. Tambahkan kode program untuk menghitung banyaknya perbandingan dan pergeseran pada algoritma pengurutan penyisipan langsung, penyisipan biner dan seleksi.
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.
4. Berikan kesimpulan dari percobaan dan latihan yang telah Anda lakukan.