2

Array 1 Dimensi

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Array 1 dimensi.*

Teori

Array adalah representas data pada lokasi memori secara berurutan dan bertipe sama. Pengaksesan data ke dalam array menggunakan indek. Karena sifatnya inilah maka array cocok digunakan untuk pengornasisasian data yang seragam.

Secara umum struktur data array telah dibuat oleh pewngembang compiler bahasa apapun termasuk bahasa pemrogram java yang mempunyai strukrut umum sebagai berikut :

typeData [] namaLarik;

Keterangan :

- typeData : bisa tipe data dasar seperti int, float dst, atau berupa tipe bentukan seperti class atau object.
- Pada dasarnya tipedata arraypun juga termasuk tipe data bentukan, sehingga atas dasar inilah kita bisa membuat varian dari applikasi array.

Contoh :

1. Int []A,B;
   - mendeklarasikan dua buah array A dan B masing-masing bertipe integer.
2. Class X{}
   X []A;
   - mendeklarasikan array A bertipe class X.

ADT Array 1 Dimensi

Namun demikian sebagai suatu struktur data array dapat kita dapat menyusun *Abstract Data Type*-nya (ADT) yang diberi nama Larik sebagai berikut :

Larik
Int size **Object []item**
BuatLarik(int) **setItem(int id, Object dt)** **int getSize()** **int getPos(Object dt)** **int []getPosPos(Object dt)** **Object getMax()** **Object getMin()** **Larik Sort();** **Larik Copy(int id, int n)**

Jika tipe data Object pada ADT di atas diganti dengan tipe data int, maka programnya adalah sebagai berikut :

```
    Program Latihan Praktikum 2.1
1   public class Larik{
2         //data (strukur data)
3         private int size;
4         private int []itemDt;
5
6         //method
7         public void buatLarik(int n){
8               this.size = n;
9               this.itemDt = new int[this.size];
10        }
11        public Larik(int n){    buatLarik(n);}
12        public int getSize(){   return this.size;}
13        public Larik(int []dt){
14              buatLarik(dt.length);
15              for      (int      i=0;      i<dt.length;      i++)
16        isiItem(i,dt[i]);
17        }
18
19        public void isiItem(int id, int dt){
20              this.itemDt[id] = dt;
21        }
22
23        public void cetak(String komentar){
24              System.out.println(komentar);
25              for(int i=0; i<this.size; i++){
26                    System.out.print(this.itemDt[i]+" ");
27              }
```

```
28              System.out.println();
29        }
30        public int findBesar(){
31              int besar = this.itemDt[0];
32              for (int i=1;i<this.size; i++){
33                    if (besar < this.itemDt[i]){
34                          besar = this.itemDt[i];
35                    }
36              }
37              return besar;
38        }
39        /**
40         * program ini mencari posisi suatu data tertentu di larik
41         */
42        public int getPosisi(int dtCari){
43              int pos = -99;
44              boolean ketemu = false;
45              int i=0;
46              while (!ketemu && i<this.size){
47                    if (dtCari == this.itemDt[i]){
48                          ketemu = true;
49                          pos = i;
50                    }
51                    i++;
52              }
53              return pos;
54        }
55
56        private int getPosMax(int id){
57              int max = this.itemDt[id];
58              int posMax = id;
59              for (int i=id+1;i<size; i++){
60                    if (max <= this.itemDt[i]) {
61                          max = this.itemDt[i];
62                          posMax = i;
63                    }
64              }
65              return posMax;
66        }
67
68        private int getPosMin(int id){
69              int min = this.itemDt[id];
70              int posMin = id;
71              for (int i=id+1;i<size; i++){
72                    if (min >= this.itemDt[i]) {
73                          min = this.itemDt[i];
74                          posMin = i;
75                    }
76              }
77              return posMin;
78        }
79        public int PencarianBiner(int dtCari,   int awal, int
80  akhir){
81              int pos = -99;
82              int tengah = (awal+akhir)/2;
```

```
83              if(dtCari< this.itemDt[tengah])
84            return PencarianBiner(dtCari, awal, tengah);
85              else if (dtCari > this.itemDt[tengah])
86              return PencarianBiner(dtCari,tengah+1,akhir);
87              else  if  (dtCari  ==  this.itemDt[tengah])  return
88    tengah;
89              else return pos;
90        }
91        /**
92         * program untuk mencopy isi suatu Larik
93         * mulai dari posisi k sebanyak n item hasilnya
94         * dikeluarkan sebagai array baru
95         */
96        public Larik copyLarik(int k, int n){
97              Larik lHasil = null;
98              if (n <= this.size-k){
99                    lHasil = new Larik(n);
100                   int j = 0;
101                   for (int i=k; i<k+n; i++){
102                         lHasil.isiItem(j++, this.itemDt[i]);
103                   }
104             }
105             return lHasil;
106       }
107
108       /**
109        * pilihan 0 : urutkan dari kecil ke besar
110        * lainya : urutkan dari besar ke kecil
111        * Algoritma pengurutan ini menggunakan selection sort
112        */
113       public Larik SelectionSort(int pilihan){
114             Larik lsort = copyLarik(0,size);
115
116             for (int i=0; i<lsort.getSize();i++){
117                   int posData;
118                   if     (pilihan     ==     0)     posData     =
119   lsort.getPosMin(i);
120                   else posData = lsort.getPosMax(i);
121
122                   int dt1 = lsort.itemDt[i];
123                   int dt2 = lsort.itemDt[posData];
124
125                   lsort.itemDt[i] = dt2;
126                   lsort.itemDt[posData] = dt1;
127
128             }
129             return lsort;
130       }
131
132       public static void main (String[] args) {
133             int []A = {2,34,5,7,10};
134             Larik lA = new Larik(A);
135
136
137             lA.cetak("Sebelum");
```

```
138             lA.Sort(0).cetak("Sesudah di sort");
139             //int k = lA.PencarianBiner(34,0,5);
140             //System.out.println(k);

                Larik lB = lA.Sort(0);
                int p = lB.PencarianBiner(10,0,5);
                System.out.println(p);
    }
}
```

Latihan 1.

1. Jalankan program di atas, amati hasilnya.
2. Bukalah tanda komentar pada 133 dan 134, jalankan dan amati hasilnya.
3. Perhatikan pada baris 131 dan 132, cobalah tambahkan perintah berikut pada baris sesudahnya `lA.cetak("Sesudah")`, amati hasilnya. Tuliskan kesimpulan anda.

Tugas 1.

1. Kembangkan program di atas dengan menambahkan method berikut :
   a. Mencari posisi bilangan yang merupakan kelipatan dari x0 yang terletak diantara index x1 dan index x2 dengan header sebagai berikut :

      ```
      int findPosKelipatan (int x0, int x1, int x2)
      ```

      Contoh :
      Sebuah set {1, 2, 3, 4, 5, 6, 7} dan dengan memanggil method findPosKelipatan(3, 1, 6) maka akan menghasilkan **2 5** karena kelipatan dari nilai 3 adalah 3 dan 6, dimana keduanya ada di antara index 1 – 6 dan memiliki indeks 2 dan indeks 5.

   b. Pengurutan dengan menggunakan algoritma pengurutan yang lain selain Selection Sort.

2. Pada Latihan kedua ini anda diminta untuk melengkapi bagian dari program ADT_Larik sehingga jika diberikan program utama pada gambar 1 akan menghasilkan keluaran sebagaimana gambar 2.

Program Latihan Praktikum 2.2

```
1  package ADT_Larik;
2  /**
3   *
4   * @author achmad ridok
5   *
6   */
7  public class Larik{
8        //data (strukur data)
9        private int size;
10       private double []itemDt;
11
12       /**
13        * Contructor untuk membuat ADT larik dari suatu array
14
15        * @param A : array bertipe int
16        */
17       public Larik(double []A){
18             this.size = A.length;
19             this.itemDt = new double[this.size];
20             for (int i=0; i<this.size; i++){
21                   this.itemDt[i] = A[i];
22             }
23       }
24       /**
25        * fungsi untu mendapatkan ukuran larik
26        * @return size dari larik
27        */
28       public int getSize(){
29             return this.size;
30       }
31
32       /**
33        * fungsi untuk mendapatkan item ke i dari suatu larik
34        * @param i : posisi item
35        * @return item larik
36        */
37       public double getItem(int i){
38             return this.itemDt[i];
39       }
40       /**
41        * fungsi static untuk menyambung dua buah larik l1 dan
42 l2
43        * @param l1 : Larik
44        * @param l2 : Larik
45        * @return Larik
46        */
47       public static Larik sambung(Larik l1, Larik l2){
48      // Lengkapi bagian ini
49       }
```

```
50
51        /**
52         * procedure untuk isiItem suatu larik
53         * @param id : indeks larik
54         * @param dt : item data yang akan disisipkan
55         */
56        public void isiItem(int id, double dt){
57              this.itemDt[id] = dt;
58        }
59
60        /**
61         * procedure cetak suatu array
62         * @param komentar : String
63         */
64        public void cetak(String komentar){
65              System.out.println(komentar);
66              for(int i=0; i<this.size; i++){
67                    System.out.printf("%.2f ",this.itemDt[i]);
68              }
69              System.out.println();
70        }
71
72        /**
73         * fungsi untuk mendapatkan nilai terbesar dari suatu
74  larik
75         * @return : item tebesar dari larik
76         */
77        public double findBesar(){
78              double besar = this.itemDt[0];
79              for (int i=1;i<this.size; i++){
80                    if (besar < this.itemDt[i]){
81                          besar = this.itemDt[i];
82                    }
83              }
84              return besar;
85        }
86        /**
87         * fungsi untuk mencari posisi suatu data tertentu di
88  array
89         * @param dtCari : data yang akan dicari
90         * @return posisiData
91         */
92        public int getPosisi(double dtCari){
93              int pos = -99;
94              boolean ketemu = false;
95              int i=0;
96              while (!ketemu && i<this.size){
97                    if (dtCari == this.itemDt[i]){
98                          ketemu = true;
99                          pos = i;
100                   }
101                   i++;
102             }
103             return pos;
104       }
```

```
105
106         /**
107          * fungsi static untuk mencopy isi suatu larik l
108          * @param k : posisi awal
109          * @param n : jumlah item yang akan dicopy
110          * @param l : larik asal
111          * @return Larik hasil copy
112          */
113         public static Larik copyLarik(int k, int n, Larik l){
114               // lenkapi bagian ini
115         }
116
117         /**
118          * fungsi untuk mencari posisi terbesar suatu data
119          * suatu posisi awal sampai akhir
120          * @param awal : posisi awal
121          * @param akhir : posisi akhir
122          * @return posisi data terbesar
123          */
124         public  int getPosBesar(int awal, int akhir){
125               int posBesar = -1;
126               double itemBesar;
127               if (awal <= akhir){
128                     posBesar = awal;
129                     itemBesar = this.getItem(awal);
130
131                     for (int i=awal+1; i<akhir; i++){
132                           double nilaiItem = this.getItem(i);
133                           if (itemBesar < nilaiItem){
134                                 itemBesar = nilaiItem;
135                                 posBesar = i;
136                           }
137                     }
138               }
139               return posBesar;
140         }
141
142         /**
143          * fungsi untuk mencari posisi data terkecil suatu array
144          * mulai dari posisi awal sampai posisi akhir
145          * @param awal : posisi awal
146          * @param akhir : posisi akhir
147          * @return posisi data terkecil
148          */
149         public  int getPosKecil(int awal, int akhir){
150               // lenkapi bagian ini
151         }
151
152         /**
153          * fungsi pengurutan suatu larik lAsal dimana kondisi
154   lAsal
155          * akan tetap setelah proses pengurutan
156          * @param lAsal : Array asal yang akan diurutkan
157          * @param status : 0-> urut dari kecil ke besar
158          *                          1-> urut dari besar ke kecil
```

```
159         * @return Array baru hasil pengurutan
160         */
161       public static Larik SelectionSort(Larik lAsal, int
162 status){
163               int n = lAsal.getSize();
164               Larik lhasil = Larik.copyLarik(0, n, lAsal);
165 
166               if (status == 0){// urutkan data dari kecil ke
167 besar
168                     for (int i=0; i<n; i++){
169                           int posKecil = lhasil.getPosKecil(i,
170 n);
171                           double itemKecil =
172 lhasil.getItem(posKecil);
173                           double itemI = lhasil.getItem(i);
174                           lhasil.isiItem(i, itemKecil);
175                           lhasil.isiItem(posKecil, itemI);
176                     }
177               } else { // urutkan data dari besar ke kecil
178                     for (int i=0; i<n; i++){
179                           int posBesar = lhasil.getPosBesar(i,
180 n);
181                           double itemBesar =
182 lhasil.getItem(posBesar);
183                           double itemI = lhasil.getItem(i);
184                           lhasil.isiItem(i, itemBesar);
185                           lhasil.isiItem(posBesar, itemI);
186                     }
187               }
188               return lhasil;
189         }
190 }
```

Gambar 1. Potongan program utama

```
public class AppPr1 {
      public static void main(String[] args) {
            // implementasi untuk ADT_Larik
            double []A = {3,4,1,10,5,2,10,20,16};
            double []B = {4,3,1,11,7};
            Larik L1 = new Larik(A);
            Larik L2 = new Larik(B);

            L1.cetak("L1");
            L2.cetak("L2");
            Larik L3 = Larik.sambung(L1, L2);
            L3.cetak("L3");
            Larik L4 = Larik.copyLarik(0, L1.getSize(), L1);
            L1.cetak("L1");
            L4.cetak("L4");

            Larik L5 = Larik.SelectionSort(L1,0);
            L5.cetak("L5");
            L1.cetak("L1");
            int []posisi = L1.FindPosPos(10);
```

```
            double hasil = Larik.LarikKaliLarik(L1, L4);
            System.out.printf("HASIL KALI %.3f\n",hasil);
      }
}
```

Dengan hasil keluaran sebagai berikut :

Gambar 2. Hasil keluaran

```
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
Isi Larik L2
4.00 3.00 1.00 11.00 7.00
L3 = gabungan dari L1 dan L2
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00 4.00 3.00 1.00 11.00
7.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L4 Copy dari L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L5 Hasil pengurutan dari L1 kecil -> besar
1.00 2.00 3.00 4.00 5.00 10.00 10.00 16.00 20.00
L6 Hasil pengurutan dari L1 besar -> kecil
20.00 16.00 10.00 10.00 5.00 4.00 3.00 2.00 1.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
HASIL KALI Larik L1*L4 = 911.000
```