Praktikum Struktur Data
Pertemuan Ke-14

Pengurutan Insertion Sort

- Metode pengurutan dengan cara menyisipkan elemen larik pada posisi yang tepat.
- Pencarian posisi yang tepat dilakukan dengan menyisir larik
- Metode pengurutan sisip cocok untuk persoalan menyisipkan elemen baru ke dalam sekumpulan elemen yang sudah terurut.

Algoritma Pengurutan sisipan untuk pengurutan menaik (Ascending)

Untuk setiap pass/langkah i = 2, ..., n
lakukan:

1. y ← L[i]
2. Sisipkan y pada tempat yang sesuai antara L[1] ... L[i]

Langkah 2

- Elemen y=L[2] harus dicari tempatnya yang tepat di dalam l[1..2] dengan cara menggeser elemen L[1..1] ke kanan (atau ke bawah) jika bayangkan larik terentang vertikal. Bila L[1..1] lebih besar daripada L[2]. Misalkan posisi yang tepat adalah k. Sisipkan L[2] pada L[k]

Langkah 3

- Elemen y= L[3] harus dicari tempatnya yang tepat di dalam L[1..3] dengan cara menggeser elemen L[1..2] ke kanan atau ke bawah) bila L[1..2] lebih besar daripada L[3]. Misalkan posisi yang tepat adalah k. Sisipkan L[3] pada L[k].

Langkah n

- Elemen y=L[n] harus dicari tempatnya yang tepat di dalam L[1..n] dengan cara menggeser L[1..n-1] ke kanan (atau ke bawah) bila L[1..n-1] lebih besar daripada L[n]. Misalkan posisi yang tepat adalah k. Sisipkan L[n] pada L[k]

Procedure Insertion Sort

```
10  Procedure Insertion_Sort(n,i,j,x : integer; var data : larikint);
11  begin
12      for i := 2 to n do
13      begin
14          x := data[i];
15          j := i-1;
16          ketemu := false;
17
18          while (j>=1) and (not ketemu) do
19          begin
20              if x < data[j] then
21              begin
22                  data[j+1] := data[j];
23                  j := j-1;
24              end
25                 else
26                 ketemu := true;
27          end;{end while}
28
29          data[j+1] := x;
30      end;
31  end;
```

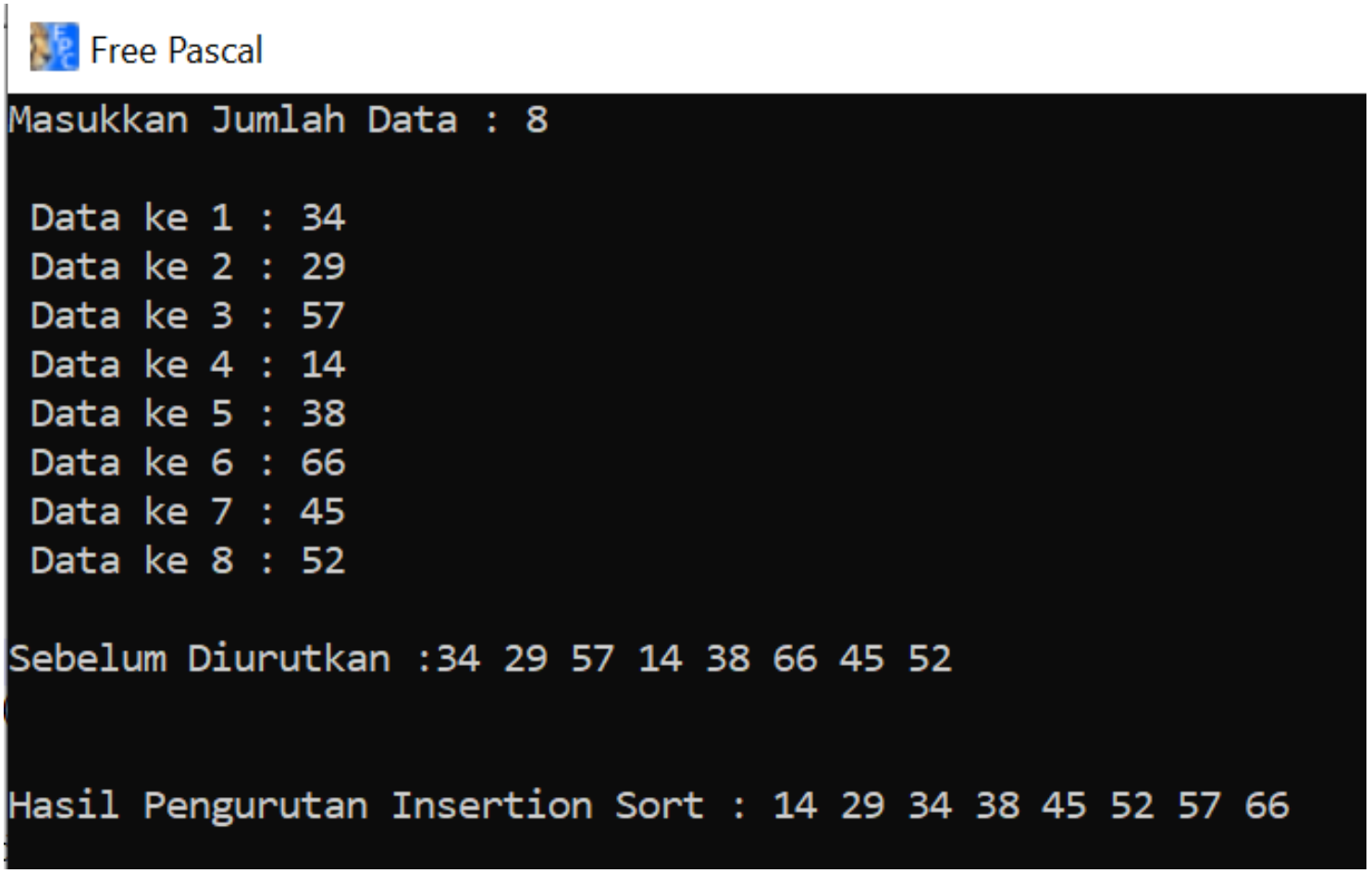
Contoh Program Insertion Sort

```
1  Program InsertSort;
2  uses crt;
3  const max = 1000;
4    type larikint = array [1..max] of integer;
5  var
6    data : larikint;
7    n,i,j,x :integer;
8    ketemu : boolean;
9
10 Procedure Insertion_Sort(n,i,j,x : integer; var data : larikint);
11 begin
12     for i := 2 to n do
13     begin
14         x := data[i];
15         j := i-1;
16         ketemu := false;
17
18         while (j>=1) and (not ketemu) do
19         begin
20             if x < data[j] then
21             begin
22                 data[j+1] := data[j];
23                 j := j-1;
24             end
25                else
26                ketemu := true;
27         end;{end while}
28
29         data[j+1] := x;
30     end;
31 end;
```

Contoh Program Insertion Sort

```
33  Begin
34  Clrscr;
35      write('Masukkan Jumlah Data : ');readln(n);
36      writeln;
37      for i := 1 to n do
38      begin
39          write (' Data Ke ',i,' : ');readln(data[i]);
40      end;
41      WriteLn;
42      write ('Sebelum Di Urutkan : ');
43      for i := 1 to n do
44          Write (data[i],' ');
45
46      Insertion_Sort(n,i,j,x,data);
47      WriteLn;
48      writeln;
49      writeln;
50      write ('Hasil Pengurutan penyisipan langsung : ');
51      for i := 1 to n do
52          write(data[i],' ');
53      ReadLn;
54  end.
```

Output Program Insertion Sort



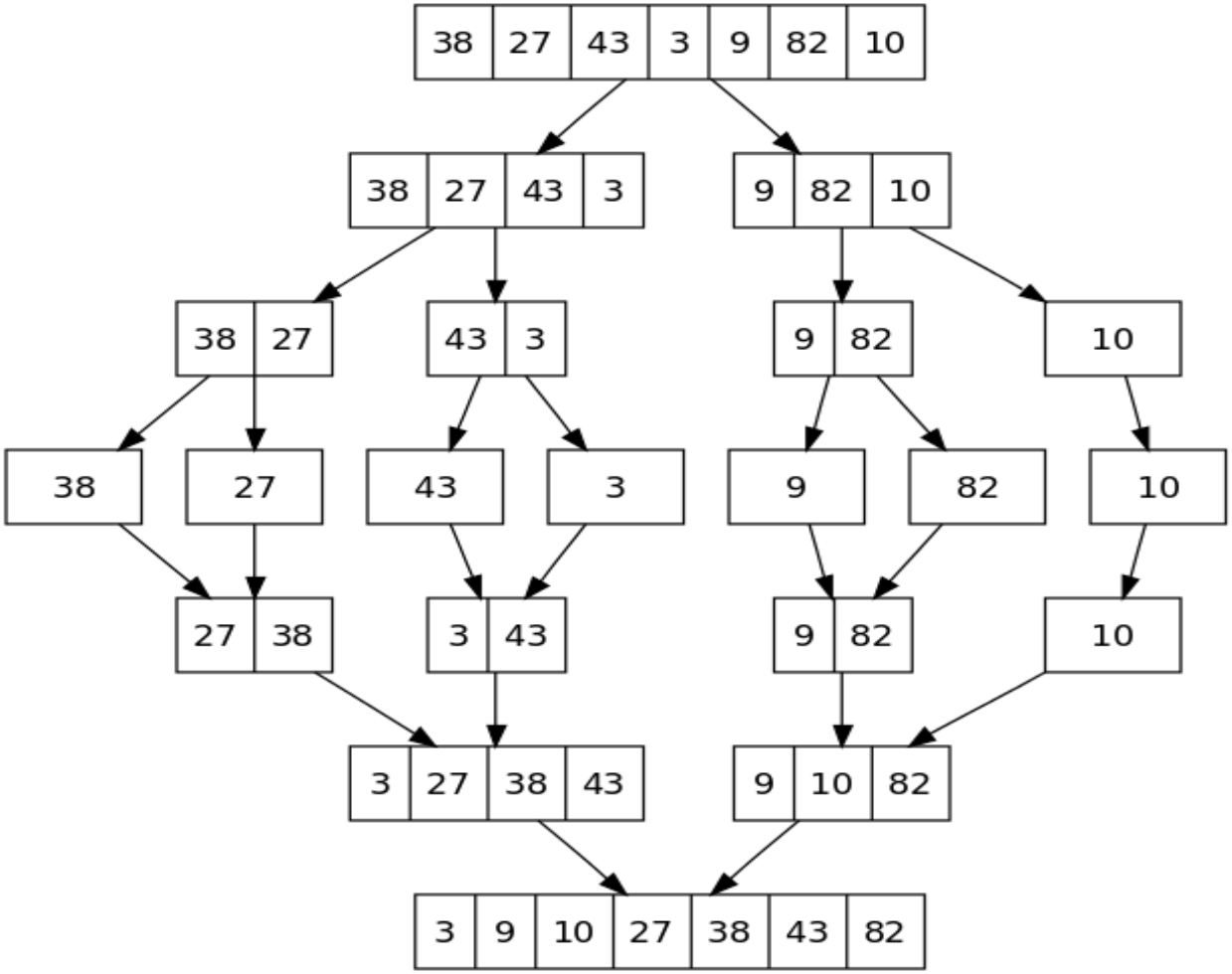
Merge Sort

- MergeSort adalah algoritma yang berdasarkan strategi divide-and-conquer.
- Algoritma ini tediri dari dua bagian utama, yaitu bagian pembagian list menjadi sublist-sublist yang lebih kecil dan bagian sort (pengurutan) dan merge (penggabungan) pada sublist-sublist tersebut.

Merge Sort

- *Divide*: membagi masalah menjadi beberapa submasalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
- *Conquer*: memecahkan (menyelesaikan) masing-masing submasalah (secara rekursif), dan
- *Combine:* mengabungkan solusi masing-masing submasalah sehingga membentuk solusi masalah semula.

Contoh Merge Sort



Cara kerja *Merge sort*

- *Divide*:

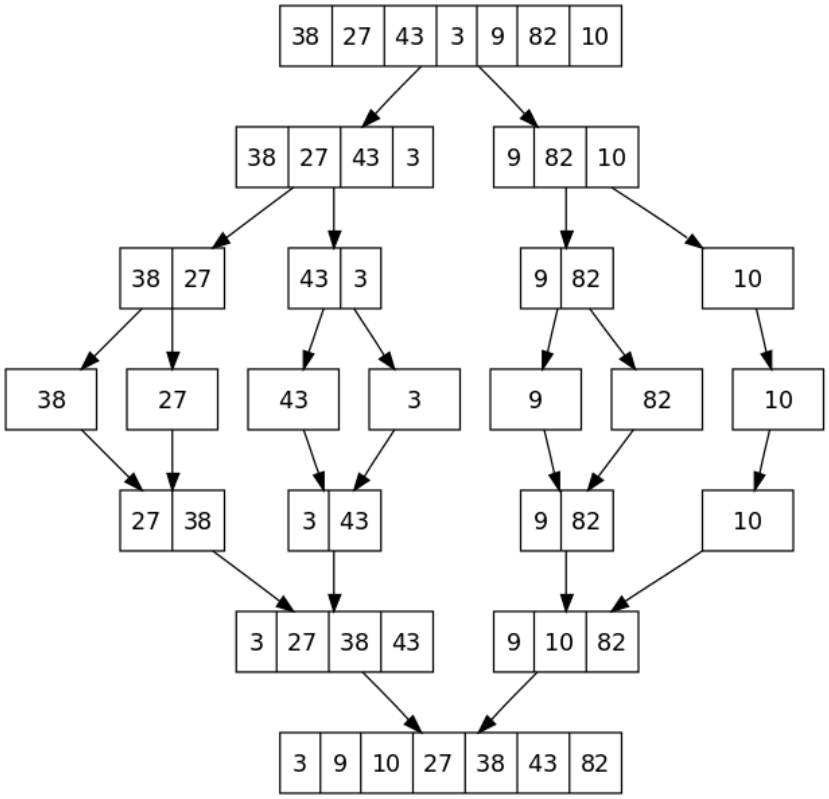
membagi sekuens dg n elemen menjadi 2 subsekuens masing-masing dgn n/2 elemen.

- *Conquer*:

mengurutkan 2 subsekuens tersebut secara rekursif menggunakan *merge sort*.

- *Combine*:

menggabungkan 2 subsekuens yang telah diurutkan untuk menghasilkan output yang dikehendaki.



Algoritma Merge Sort

```
procedure mergesort(input/output a : tabelint, input i, j : integer)
{ mengurutkan tabel L[i..j] dengan algoritma merge sort masukan: tabel L dengan n
    elemen
keluaran: tabel a yang terurut }
deklarasi:
    k : integer
algoritma:
    if i < j then { ukuran(L)> 1}
    k←(i+j) div 2
    mergesort(L, i, k)
    mergesort(L, k+1, j)
    merge(L, i, k, j)
    endif
    endprocedure
```

Contoh Program Merge Sort

```
1  Program ContohMergeSort;
2  uses crt;
3  type arr = array [1..100] of integer;
4
5  var
6  ArrMain,ArrUrut : arr;
7  n,m : integer;
8
```

```
Procedure Merge(Left:arr; pjgL:integer; Right : arr; pjgR:integer; var ArrMerge : arr);
var
    i,j,k,m,panjang: integer;
    hasil : arr;
begin
    i:=1;
    j:=1;
    k:=1;
    panjang:=pjgL+pjgR;
    while ((pjgL>0) and (pjgR>0)) do
    begin
        if(Left[i]<= Right[j]) then
        begin
            hasil[k]:=Left[i];
            i:=i+1;
            k:=k+1;
            pjgL:=pjgL-1;
        end
        else
        begin
            hasil[k]:=Right[j];
            j:=j+1;
            k:=k+1;
            pjgR:=pjgR-1;
        end;
    end;

```

```
36      while (pjgL>0) do
37      begin
38          hasil[k]:=Left[i];
39          i:=i+1;
40          k:=k+1;
41          pjgL:=pjgL-1;
42      end;
43
44      while (pjgR>0) do
45      begin
46          hasil[k]:=Right[j];
47          j:=j+1;
48          k:=k+1;
49          pjgR:=pjgR-1;
50      end;
51
52      {Mengembalikan nilai ke variabel output ArrMerge}
53      ArrMerge:=hasil;
54      for m:= 1 to panjang do
55      writeln('Array Merge ke-',m,' : ',hasil[m]);
56  end;
```

```
58 Procedure Mergesort(pjg:integer;A : arr;var ArrHasil : arr);
59 var
60     middle,i,pjgLeft,pjgRight : integer;
61     ArrLeft,ArrRight : arr;
62 begin
63     if pjg <= 1 then
64         ArrHasil := A
65     else
66     begin
67         middle := pjg div 2;
68         for i:=1 to middle do
69             ArrLeft[i]:=A[i];
70
71         for i:=(middle+1) to pjg do
72             ArrRight[i-middle]:=A[i];
73             pjgLeft := pjg div 2;
74             pjgRight := (pjg+1) div 2;
75
76         for m:= 1 to pjgLeft do
77             writeln('Array Left ke-',m,' : ',ArrLeft[m]);
78
79         for m:= 1 to pjgRight do
80             writeln('Array Right ke-',m,' : ',ArrRight[m]);
81             mergesort(pjgLeft,ArrLeft,ArrLeft);
82             mergesort(pjgRight,ArrRight,ArrRight);
83             merge(ArrLeft,pjgLeft,ArrRight,pjgRight,ArrHasil);
84     end;
85 end;
```

```
87  begin
88  clrscr;
89  write('Jumlah array : ');readln(n);
90      for m := 1 to n do
91      begin
92          write('Array ke-',m,' : ');
93          readln(ArrMain[m]);
94      end;
95  writeln;
96  writeln('LANGKAH MERGE SORT');
97  writeln('--------------------');
98  mergesort(n,ArrMain,ArrUrut);
99  writeln;
100 writeln('HASIL ARRAY TERURUT');
101 writeln('----------------------');
102 for m:= 1 to n do
103 writeln('Array Urut ke-',m,' : ',ArrUrut[m]);
104 readln;
105 end.
```

Output Program Merge Sort

Free Pascal

```
Jumlah Array : 7
Array ke-1 : 38
Array ke-2 : 27
Array ke-3 : 43
Array ke-4 : 3
Array ke-5 : 9
Array ke-6 : 82
Array ke-7 : 10
```

Free Pascal

```
Array Merge ke-2 : 68
Array Left ke-1 : 3
Array Right ke-1 : 9
Array Merge ke-1 : 3
Array Merge ke-2 : 9
Array Left ke-1 : 82
Array Right ke-1 : 10
Array Merge ke-1 : 10
Array Merge ke-2 : 82
Array Merge ke-1 : 3
Array Merge ke-2 : 9
Array Merge ke-3 : 10
Array Merge ke-4 : 82
Array Merge ke-1 : 3
Array Merge ke-2 : 9
Array Merge ke-3 : 10
Array Merge ke-4 : 27
Array Merge ke-5 : 38
Array Merge ke-6 : 43
Array Merge ke-7 : 82
```

```
 Hasil Array Terurut
--------------------
Array urut ke-1 : 3
Array urut ke-2 : 9
Array urut ke-3 : 10
Array urut ke-4 : 27
Array urut ke-5 : 38
Array urut ke-6 : 43
Array urut ke-7 : 82
```

Quick Sort

- Mengurutkan dengan membandingkan suatu elemen (pivot) dengan elemen yang lain dan menyusun sedemikian rupa sehingga elemen-elemen lainnya yang lebih kecil daripada pivot tersebut terletak di sebelah kirinya dan elemen-elemen lain yang lebih besardaripada pivot terletak di sebelah kanannya

Quick Sort

- Metode Quick sering disebut juga metode partisi (partition exchange sort).
- Metode ini mempunyai efektifitas yang tinggi dengan teknik menukarkan dua elemen dengan jarak yang cukup besar.

Quick Ascending

```
Procedure Asc_quick(L,R : integer); { Prosedur ascending }
Var
  i, j : integer;
  begin
     if L < R then
     begin
    i := L ; j := R+1;
    repeat
       repeat inc(i) until data[ i ] >= data [ l ];
       repeat dec(j) until data[ j] <= data [ l ];
       if i< j then TukarData (data[i], data[j]);
    until i > j;
    TukarData(data[ l ], data [ j ]);
    Asc_quick(L, j-1);
       Asc_quick(j+1, R);
     End;
  End;
```

Quick Descending

```
Procedure Desc_quick(L,R : integer); { Prosedur Descending }
Var
  i, j : integer;
  begin
     if L < R then
     begin
    i := L ; j := R+1;
    repeat
       repeat inc(i) until data[ i ] <= data [ l ];
       repeat dec(j) until data[ j] >= data [ l ];
       if i< j then TukarData (data[i], data[j]);
    until i > j;
    TukarData(data[ l ], data [ j ]);
    Asc_quick(L, j-1);
    Asc_quick(j+1, R);
     End;
End;
```

Procedure Quick Sort dengan nilai tengah sebagai pembanding (pivot)

```
Procedure asc_Quick(L,R : integer); { Prosedur Ascending}
Var
  Mid, i, j : integer;
  Begin
     i := L ;  j := R ;  mid := data [(L+R) div 2] ;
     repeat
    while data [ i ] < mid do inc(i);
    while data [ j ] > mid do dec(j);
    if i <= j then
       begin
      Change(data[ i ], data[ j ]);
      Inc(i); dec(j);
       End;
    Until i > j;
    If L < j then Asc_Quick( L, j );
    If i < R then Asc_Quick(i, R);
  End;
```

```
Procedure desc_Quick(L,R : integer); { Prosedur Descending}
Var
  Mid, i, j : integer;
  Begin
     i := L ;  j := R ;  mid := data [(L+R) div 2] ;
     repeat

    while data [ i ] > mid do inc(i);
    while data [ j ] < mid do dec(j);
    if i <= j then
       begin
      Change(data[ i ], data[ j ]);
      Inc(i); dec(j);
       End;
     Until i > j;
     If L < j then Asc_Quick( L, j );
     If i < R then Asc_Quick(i, R);

  End;
```

Tugas

Buatlah program pengurutan dengan menggunakan metode Selection Maximum (Descending)


Thank you!