JURNAL PRAKTIKUM
(LAB. ACTIVITY)

STRUKTUR DATA
SI025

Materi 4 dan 5:
SORTING
(PENGURUTAN)

Dosen:

Agung Nugroho, M.Kom
Prof. Dr. Ema Utami, S.Si, M.Kom
Ikmah, M.Kom
Ninik Tri H, M.Kom
Lilis Dwi Farida, M.Kom
Windha Mega PD, M.Kom

S1 – SISTEM INFORMASI
UNIVERSITAS AMIKOM YOGYAKARTA
2018

Sorting(Pengurutan)
Pendahuluan

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami apa itu Sorting
2. Memahami jenis dari metode sorting (buble sort,selection sort,insertion sort)
3. Menerapkan Sorting menggunakan bahasa C++

B. Peralatan

1. PC Desktop
2. Windows 7
3. MinGW

C. Teori

SORTING (PENGURUTAN)

Sorting adalah proses mengatur sekumpulan objek menurut aturan atau susunan tertentu.

Pengurutan dapat dilakukan secara ascending (urut naik = dari data kecil ke data lebih besar) dan descending (urut turun = dari data besar ke data lebih kecil)

Banyak sekali algoritma pengurutan yang ada, tetapi disini akan kita pelajari 3 metode yaitu:

- Pengurutan Gelembung (bubble sort)
- Maksimum/Minimum (maksimum/minimum sort)
- Pengurutan sisip (insertion Sort)

Contoh: Data Acak : 5 6 8 1 3 25 10

- Ascending : 1 3 5 6 8 10 25
- Descending : 25 10 8 6 5 3 1

I. PENGURUTAN GELEMBUNG (BUBLE SORT)

Metode pengurutan gelembung (bubble sort) diinspirasi oleh gelembung sabun yang ada di permukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air maka gelembung sabun akan selalu mengapung. Prinsip pengapungan ini juga dipakai pada pengurutan gelembung. Elemen yang berharga paling kecil "diapungkan", artinya diangkat ke atas (atau ke ujung paling kiri) melalui pertukaran. Proses pengapungan ini dilakukan N kali langkah. Pada langkah ke-I, Larik[1..N] akan terdiri dari 2 bagian yaitu:

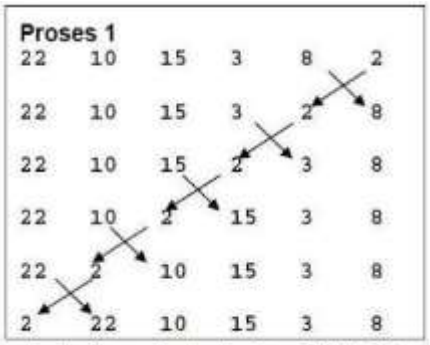
- Bagian yang sudah terurut yaitu L[1]..L[i].
- Bagian yang belum terurut L[i+1]..L[n].

Aturan buble sort(gelembung)

- mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- Jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar, jika pengurutan ascending.
- Jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar, jika pengurutan descending.
- Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan
-

Contoh : Apabila di dalam array dipunyai 6 buah data yang belum terurut yaitu 22, 10, 15, 3, 8, 2. Langkah pengurutan menggunakan metode Bubble Sort adalah sebagai berikut :

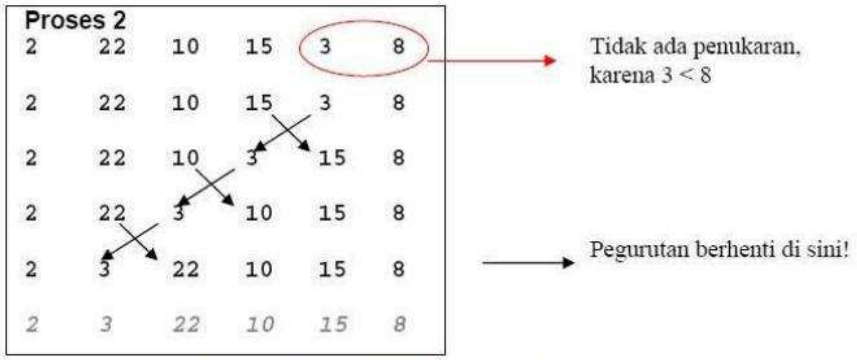
Proses 1



Gambar 1. Proses ke-1 algoritma Bubble Sorting

Pada gambar 1, pegecekan dimulai dari data yang paling akhir, kemudian dibandingkan dengan data di depannya, jika data di depannya lebih besar maka akan ditukar.

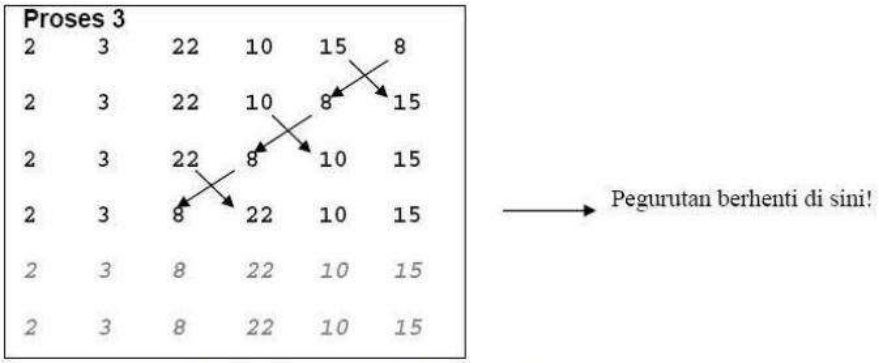
Proses 2



Gambar 2. Proses ke-2 algoritma Bubble Sorting

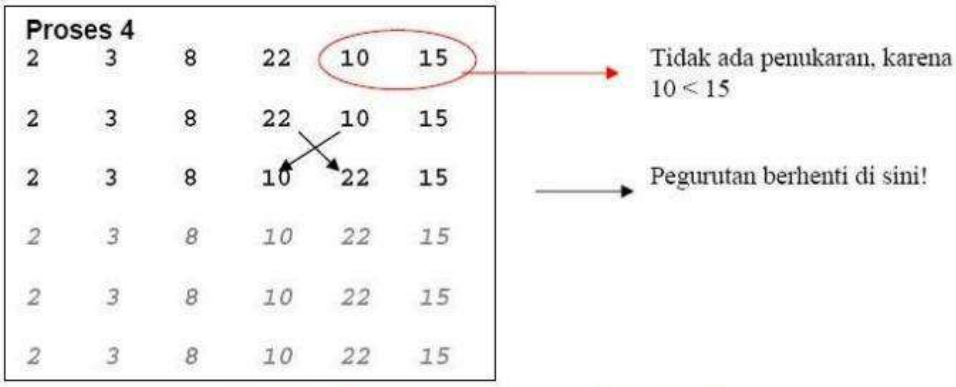
Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.

Proses 3



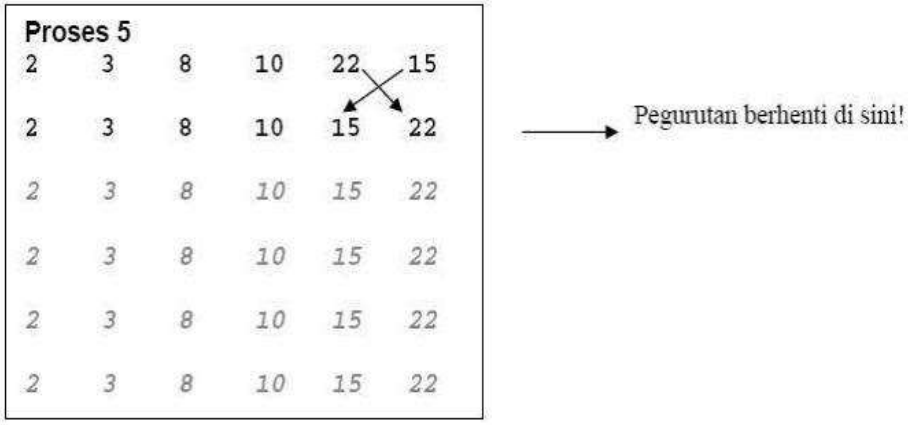
Gambar 3. Proses ke-3 algoritma Bubble Sorting

Proses 4



Gambar 4. Proses ke-4 algoritma Bubble Sorting

Proses 5



Gambar 5. Proses ke-5 algoritma Bubble Sort

D.I Praktikum Buble sort

Latihan

Kode 4.1. ketikkan program pengurutan secara Ascending seperti di bawah ini:

```
#include <iostream>

using namespace std;
int main(){
int number[5]={3,5,2,7,4};
int min=0;

//menampilkan isi array sebelum diurutkan
cout<<"Sebelum Urut: ";
for (int i=0;i<5;i++){
cout<<number[i]<<" ";
}

for (int i=1;i<5;i++){ //perulangan sejumlah data - 1
    for (int j=4;j>=i;j--) //perulangan untuk mencari nilai terkecil
        {if(number[j]<number[j-1])
            {min=number[j];
            number[j]=number[j-1];
            number[j-1]=min;}
        }
}

//menampilkan isi array setelah diurutkan
cout<<"\n\nSetelah Urut: ";
for (int i=0;i<5;i++){
cout<<number[i]<<" ";
}
return 0;
}
```

Output kode 4.1:

```
Sebelum Urut: 3 5 2 7 4

Setelah Urut: 2 3 4 5 7
```

Kode 4.2. Program pengurutan secara Descending

```
#include <iostream>

using namespace std;
int main(){
int number[5]={3,5,2,7,4};
int max=0;

//menampilkan isi array sebelum diurutkan
cout<<"Sebelum Urut: ";
for (int i=0;i<5;i++){
cout<<number[i]<<" ";
}

for (int i=1;i<5;i++){ //perulangan sejumlah data - 1
    for (int j=4;j>=i;j--) //perulangan untuk mencari nilai terbesar
        {if(number[j]>number[j-1])
            {max=number[j];
            number[j]=number[j-1];
            number[j-1]=max;}
        }
}

//menampilkan isi array setelah diurutkan
cout<<"\n\nSetelah Urut: ";
for (int i=0;i<5;i++){
cout<<number[i]<<" ";
}
return 0;
}
```

Output kode 4.2

```
Sebelum Urut: 3 5 2 7 4

Setelah Urut: 7 5 4 3 2
```

Kesimpulan Bubble Sort:

Pengurutan dengan buble sort ini kurang efisien karena terlalu banyak penukaran yang dilakukan pada setiap langkah dan membutuhkan banyak waktu serta proses lebih lama sehingga tidak direkomendasikan untuk digunakan, akan tetapi metode ini mudah dipahami dan sederhana.

Penugasan Buble Sort!

1. Buat program untuk mengurutkan huruf dengan inputan dinamis secara ascending dan descending.

II. SELECTION SORT(MAKSIMUM/MINIMUM)

Metode seleksi melakukan pengurutan dengan cara mencari data yang terkecil /terbesar untuk kemudian menukarkannya dengan data yang digunakan sebagai acuan atau sering dinamakan pivot.

Terdapat dua pendekatan dalam metode pengurutan dengan Selection Sort :

1. Algoritma pengurutan maksimum (*maximum selection sort*), yaitu memilih elemen maksimum sebagai basis pengurutan.
2. Algoritma pengurutan minimum (*minimum selection sort*), yaitu memilih elemen minimum sebagai basis pengurutan.

Contoh :

Tinjau larik dengan N=6 buah elemen dibawah ini yang belum terurut menjadi diurut naik.

29	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

Cari elemen maksimum di dalam larik L[1..6]→ maks = L[5] = 76

Tukar maks dengan L[N],hasil akhir langkah 1:

29	27	10	8	21	76
1	2	3	4	5	6

Langkah 2:

(berdasarkan susunan larik hasil langkah 1)

Cari elemen maksimum di dalam larik L[1..5]→ maks = L[1] = 29

Tukar maks dengan L[5],hasil akhir langkah 2:

21	27	10	8	29	76
1	2	3	4	5	6

Langkah 3:

(berdasarkan susunan larik hasil langkah 2)

Cari elemen maksimum di dalam larik L[1..4]→ maks = L[2] = 27

Tukar maks dengan L[4],hasil akhir langkah 3:

21	8	10	27	29	76
1	2	3	4	5	6

Langkah 4:

(berdasarkan susunan larik hasil langkah 3)

Cari elemen maksimum di dalam larik L[1..3]→ maks = L[1] = 21

Tukar maks dengan L[3],hasil akhir langkah 4:

10	8	21	27	29	76
1	2	3	4	5	6

Langkah 5:

(berdasarkan susunan larik hasil langkah 4)

Cari elemen maksimum di dalam larik L[1..2]→ maks = L[1] = 10

Tukar maks dengan L[2],hasil akhir langkah 5:

8	10	21	27	29	76
1	2	3	4	5	6

Selesai. Larik sudah terurutkan !

Untuk algoritma Pengurutan Minimum caranya sama persis dengan maksimum hanya saja yang ditukar adalah nilai yang minimum bukan maksimum.

D.II. Praktikum Selection sort

Kode 4.3. ketikkan program pengurutan selection sort seperti di bawah ini:

```
using namespace std;

int main(){
//deklarasi array dengan 5 elemen
int A[5];
int j,k,i,temp;
int jmax,u=4;

//memasukkan nilai sebelum diurutkan
cout<<"Masukkan nilai pada elemen array :"<<endl;
for(i=0;i<5;i++)
{ cout<<"A["<<i<<"]=";
 cin>>A[i];}

//menampilkan nilai sebelum diurutkan
cout<<"\nNilai elemen Array sebelum diurutkan ="<<endl;
for(i=0;i<5;i++)
{ cout<<A[i]<<"   ";}
cout<<"\n";
```

```
//Proses pengurutan secara menaik (Ascending)
for(j=0;j<5;j++)
{ jmax=0;
  for(k=1;k<=u;k++)
  if (A[k] > A[jmax])
     jmax=k;

  temp=A[u];
  A[u]=A[jmax];
  A[jmax]=temp;
  u--;

 cout<<"Hasil Proses ke-"<<j+1<<" =  ";
 for (k=0;k<5;k++)
     cout<<A[k]<<"  ";
     cout<<endl;}

//menampilkan nilai setelah diurutkan
cout<<"\nNilai setelah diurutkan ="<<endl;
for(i=0;i<5;i++)
{ cout<<A[i]<<"   ";}
getch();}
```

Output kode 4.3

```
C:\Windows\system32\cmd.exe - 1
D:\sd2017>g++ -o 1 sort2.cpp

D:\sd2017>1
Masukkan nilai pada elemen array :
A[0]=13
A[1]=7
A[2]=5
A[3]=9
A[4]=10

Nilai elemen Array sebelum diurutkan =
13   7   5   9   10
Hasil Proses ke-1 =  10  7  5  9  13
Hasil Proses ke-2 =  9  7  5  10  13
Hasil Proses ke-3 =  5  7  9  10  13
Hasil Proses ke-4 =  5  7  9  10  13
Hasil Proses ke-5 =  5  7  9  10  13

Nilai setelah diurutkan =
5   7   9   10   13
```

Kesimpulan Selection sort.

dibandingkan dengan pengurutan gelembung (bubble sort) pengurutan dengan metode selection sort (maksimum/minimum) ini memiliki kinerja yang lebih baik. Operasinya pertukaran hanya sekali saja dilakukan pada setiap langkah sehingga waktu pengurutan dapat lebih ditekan. ***Metode ini direkomendasikan untuk dipakai.***

III. PENGURUTAN SISIP(INSERTION SORT)

Dari namanya, pengurutan sisip (insertion sort) adalah metode pengurutan dengan cara menyisipkan elemen larik pada posisi yang tepat. Pencarian posisi yang tepat dilakukan dengan pencarian beruntun. Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil,maka akan ditempatkan (di insert) diposisi yang seharusnya.

Proses 1:

Proses 1

0	1	2	3	4	5
22	**10**	15	3	8	2

Temp	Cek	Geser
10	Temp<22?	Data ke-0 ke posisi 1

Temp menempati posisi ke -0

0	1	2	3	4	5
10	22	15	3	8	2

Proses 2:

Proses 2

0	1	2	3	4	5
10	22	**15**	3	8	2

Temp	Cek	Geser
15	Temp<22	Data ke-1 ke posisi 2
15	Temp>10	-

Temp menempati posisi ke-1

0	1	2	3	4	5
10	**15**	22	3	8	2

Proses 3:

Proses 3

0	1	2	3	4	5
10	15	22	**3**	8	2

Temp	Cek	Geser
3	Temp<22	Data ke-2 ke posisi 3
3	Temp<15	Data ke-1 ke posisi 2
3	Temp<10	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
3	10	15	22	8	2

Proses 4:

Proses 4

0	1	2	3	4	5
3	10	15	22	**8**	2

Temp	Cek	Geser
8	Temp<22	Data ke-3 ke posisi 4
8	Temp<15	Data ke-2 ke posisi 3
8	Temp<10	Data ke-1 ke posisi 2
8	Temp>3	-

Temp menempati posisi ke-1

0	1	2	3	4	5
3	**8**	10	15	22	2

Proses 5:

Proses 5

0	1	2	3	4	5
3	8	10	15	22	**2**

Temp	Cek	Geser
2	Temp<22	Data ke-4 ke posisi 5
2	Temp<15	Data ke-3 ke posisi 4
2	Temp<10	Data ke-2 ke posisi 3
2	Temp<8	Data ke-1 ke posisi 2
2	Temp<3	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
2	3	8	10	15	22

D. Praktikum Selection sort

Kode4.4.

```
main()
{
int j,k, temp;
int L[5];

L[1]=1;
L[2]=25;
L[3]=10;
L[4]=30;
L[5]=2;
for(k=2;k<=5;k++)
{
 temp=L[k];/* ambil elemen L[k] supaya tidak tertimpa
penggeseran*/
 /* Cari Posisi Yang tepat dalam L[1..k-1] sambil menggeser*/
 j=k-1;
    while(temp<=L[j])
    {
    L[j+1]=L[j];
    j--;
    }
  if((temp >= L[j])|| ( j=1))
  L[j+1]=temp; /*posisi yg tepat untuk L[k] ditemukan*/
  else
  {
  L[j+1]=L[j];
  L[j]=temp;
  }
}

for(k=1;k<=5;k++)
 printf("%d\n", L[k]);
getch();}
```

Output kode 4.4.

```
C:\Windows\system32\cmd.exe - 1
D:\sd2017>g++ -o 1 sort3.cpp

D:\sd2017>1
1
2
10
25
30
```

E. Tugas

Buat lah program untuk mengurutkan nama teman sekelas anda.

Langkah :

1. Inputkan data secara dinamis, minimal 5 data
2. Buatlah pilihan pengurutan secara ascending dan decending
3. Tampilkan hasil sebelum pengurutan dan sesudah pengurutan

Contoh output tugas:

Pengurutan Ascending

```
C:\Windows\system32\cmd.exe - 1
Data sebelum diurutkan secara Ascending
Nama ke-1 = bayu
Nama ke-2 = angga
Nama ke-3 = rini
Nama ke-4 = dewi
Nama ke-5 = kevin

Data setelah diurutkan secara Ascending
Nama ke-1 = angga
Nama ke-2 = bayu
Nama ke-3 = dewi
Nama ke-4 = kevin
Nama ke-5 = rini
```

Pengurutan Descending

```
C:\Windows\system32\cmd.exe - 1
Data sebelum diurutkan secara Descending
Nama ke-1 = bayu
Nama ke-2 = Angga
Nama ke-3 = rini
Nama ke-4 = dewi
Nama ke-5 = kevin

Data setelah diurutkan secara Descending
Nama ke-1 = rini
Nama ke-2 = kevin
Nama ke-3 = dewi
Nama ke-4 = bayu
Nama ke-5 = Angga
```