BAB 2
SORTING (PENGURUTAN)

1. Tujuan Instruksional Umum

a. Mahasiswa mampu melakukan perancangan aplikasi menggunakan Struktur Sorting (pengurutan)
b. Mahasiswa mampu melakukan analisis pada algoritma Sorting yang dibuat
c. Mahasiswa mampu mengimplementasikan algoritma Sorting pada sebuah aplikasi secara tepat dan efisien

2. Tujuan Instruksional Khusus

a. Mahasiswa mampu menjelaskan mengenai algoritma Sorting
b. Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma Sorting
c. Mahasiswa mampu menerapkan dan mengimplementasikan algoritma Sorting

Pendahuluan

Pengurutan data dalam struktur data sangat penting terutama untuk data yang beripe data numerik ataupun karakter. Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun). Pengurutan (Sorting) adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu.

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

Deklarasi Array Sorting

Mendeklarasikan array secara global:

```
int  data[100];
int  n;  //untuk  jumlah  data
```

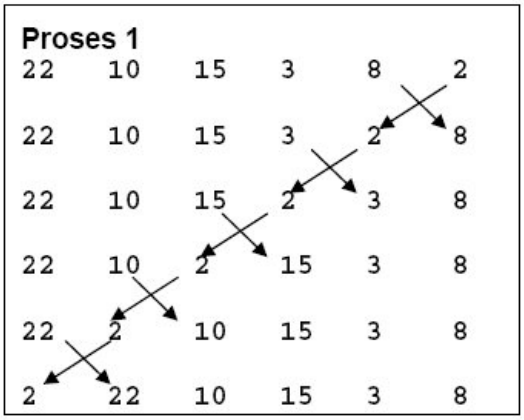
Fungsi Tukar 2 Buah Data:

```
void  tukar(int  a,int  b){
      int  tmp;
      tmp  =  data[a];
      data[a]  =  data[b];
      data[b]  =  tmp;
}
```

BUBBLE SORT

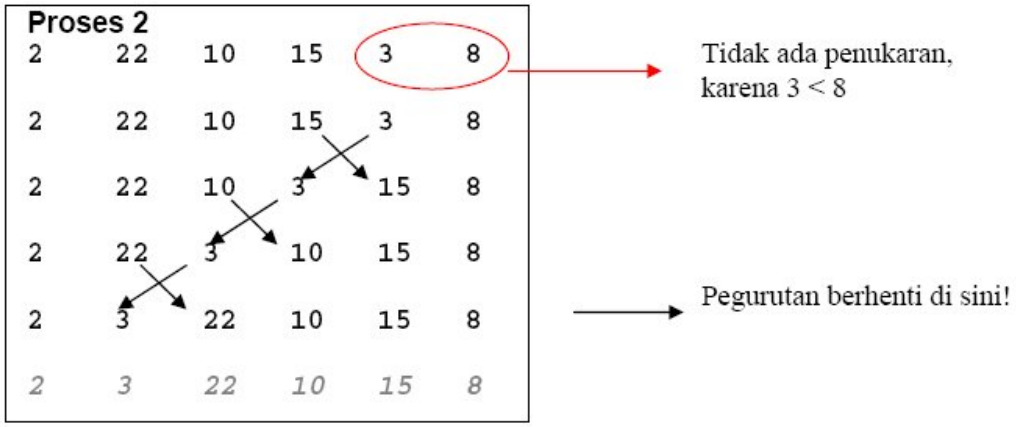
Merupakan metode sorting termudah, diberi nama "Bubble" karena proses pengurutan secara berangsur- angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda. Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya. 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. Algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya. Ketika satu proses telah selesai, maka bubble sort akan

mengulangi proses, demikian seterusnya. Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.



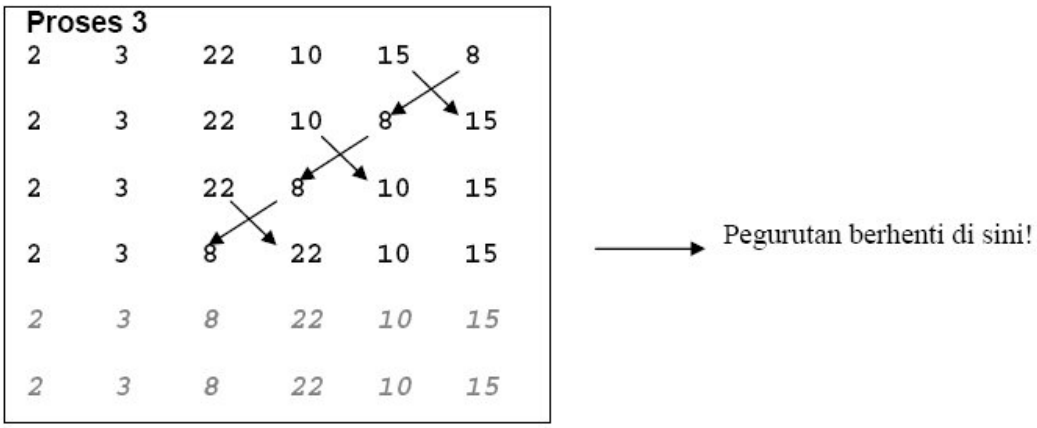
Gambar 1. Proses ke-1 algoritma Bubble Sorting

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

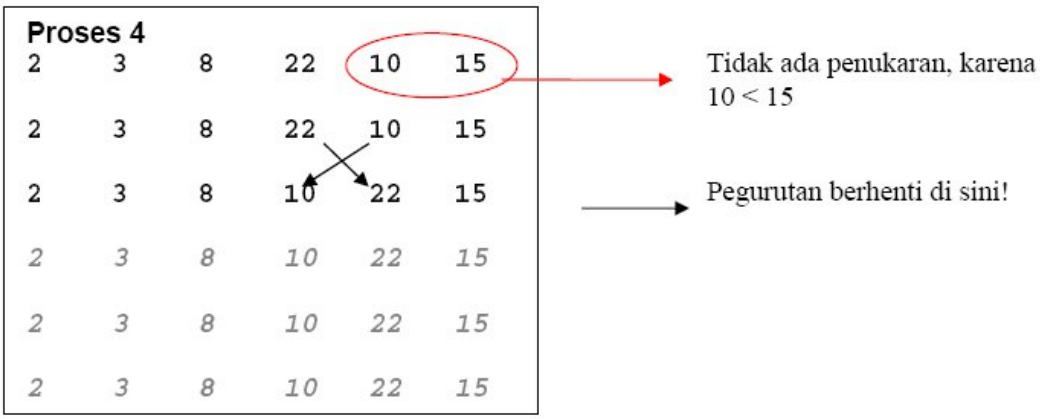


Gambar 2. Proses ke-2 algoritma Bubble Sorting

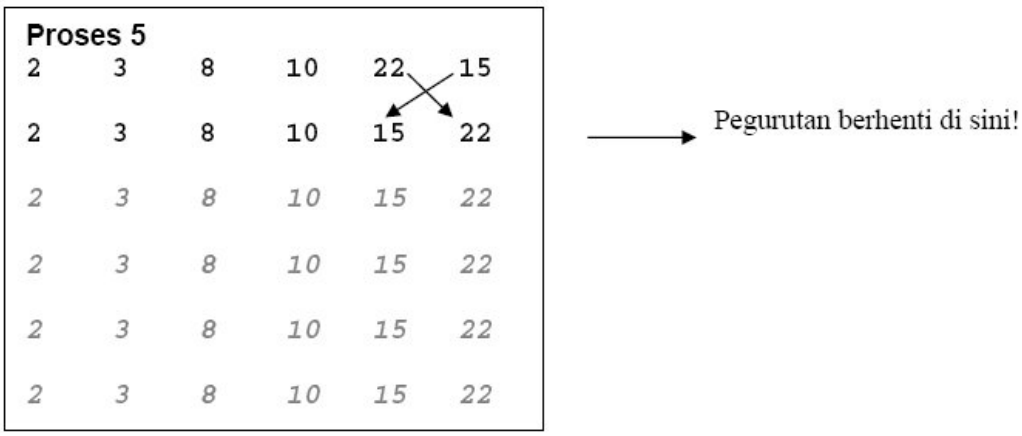
Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.



Gambar 3. Proses ke-3 algoritma Bubble Sorting



Gambar 4. Proses ke-4 algoritma Bubble Sorting



Gambar 5. Proses ke-5 algoritma Bubble Sort

Sintaks program fungsi Bubble Sort

```
void  bubble_sort(){
      for(int  i=1;i<n;i++){
            for(int  j=n-1;j>=i;j--){
                  if(data[j]<data[j-1])
```

```
                                tukar(j,j-1);   //ascending
                    }
                }
            }
```

Dengan prosedur diatas, data terurut naik (ascending), untuk urut turun (descending) silahkan ubah bagian:

```
if  (data[j]<data[j-1])
      tukar(j,j-1);
```

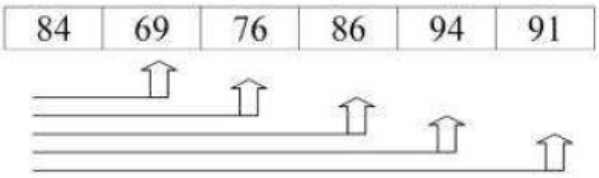
Menjadi:

```
if  (data[j]>data[j-1])
      tukar(j,j-1);
```

Algoritma Bubble Sorting mudah dalam sintaks, tetapi lebih lambat dibandingkan dengan algoritma sorting yang lain

EXCHANGE SORT

Sangat mirip dengan Bubble Sort, dan banyak yang mengatakan Bubble Sort sama dengan Exchange Sort. Pebedaan ada dalam hal bagaimana membandingkan antar elemen-elemennya. Exchange sort membandingkan suatu elemen dengan elemen-elemen lainnya dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen pusat (pivot). Sedangkan Bubble sort akan membandingkan elemen pertama/terakhir dengan elemen sebelumnya/sesudahnya, kemudian elemen sebelum/sesudahnya itu akan menjadi pusat (pivot) untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.



Proses 1

Pivot (Pusat)

84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Proses 2

Pivot (Pusat)

94	**69**	**76**	84	86	91
94	**76**	**69**	**84**	86	91
94	**84**	69	**76**	**86**	91
94	**86**	69	76	**84**	**91**
94	**91**	69	76	84	86

Proses 3

Pivot (Pusat)

94	91	**69**	**76**	84	86
94	91	**76**	**69**	**84**	86
94	91	**84**	69	**76**	**86**
94	91	**86**	69	76	84

Proses 4

Pivot (Pusat)

94	91	86	**69**	**76**	84
94	91	86	**76**	**69**	**84**
94	91	86	**84**	69	**76**

Proses 5

Pivot (Pusat)

94	91	86	84	**69**	**76**
94	91	86	84	**76**	**69**

Gambar 6. Proses algoritma Exchange Sorting

Sintaks program fungsi Exchange Sort

```
void  exchange_sort()
{
      for  (int  i=0;  i<n-1;  i++){
            for(int  j  =  (i+1);  j<n;  j++){
                  if  (data  [i]  <  data[j])
                        tukar(i,j);  //descending
            }
      }
}
```

SELECTION SORT

Merupakan kombinasi antara sorting dan searching. Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array. Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1]). Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses.

Proses 1

0	1	2	3	4	5
32	75	69	58	21	40

Pembanding	Posisi
32 < 75	0
32 < 69	0
32 < 58	0
32 > 21 (**tukar idx**)	4
21 < 40	4

Tukar data ke-0 (**32**) dengan data ke-4 (**21**)

0	1	2	3	4	5
21	75	69	58	32	40

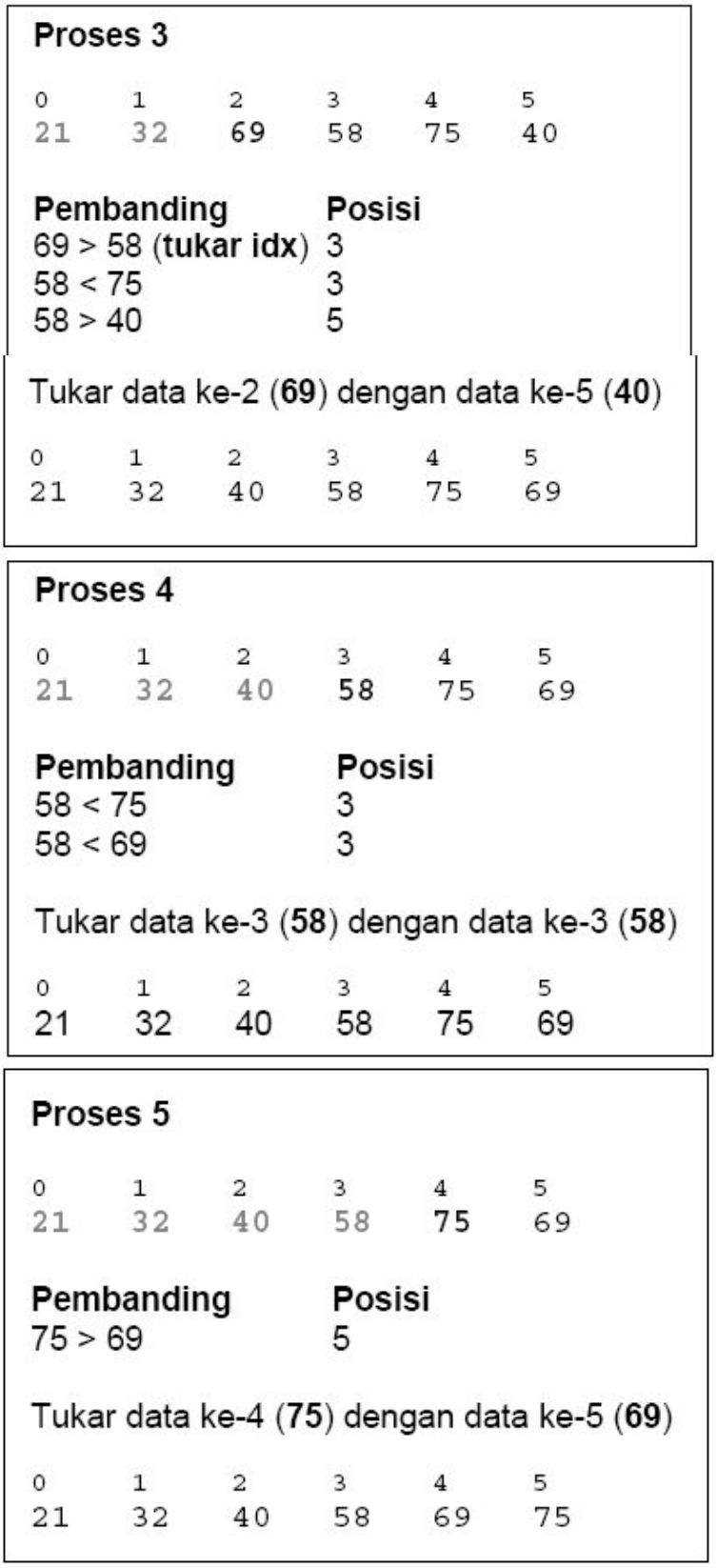
Proses 2

0	1	2	3	4	5
21	75	69	58	32	40

Pembanding	Posisi
75 > 69 (**tukar idx**)	2
69 > 58 (**tukar idx**)	3
58 > 32 (**tukar idx**)	4
32 < 40	4

Tukar data ke-1 (**75**) dengan data ke-4 (**32**)

0	1	2	3	4	5
21	32	69	58	75	40



Gambar 7. Proses algoritma Selection Sorting

Sintaks program fungsi Selection Sort

```
void  selection_sort(){
      for(int  i=0;i<n-1;i++){
            pos  =  i;
            for(int  j=i+1;j<n;j++){
                  if(data[j]  <  data[pos])
                        pos  =  j; //ascending
            }
            if(pos  !=  i)  tukar(pos,i);
      }
}
```

INSERTION SORT

Mirip dengan cara orang mengurutkan kartu, selembar demi selembar kartu diambil dan disisipkan (insert) ke tempat yang seharusnya. Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan (diinsert) diposisi yang seharusnya. Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang.

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

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

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

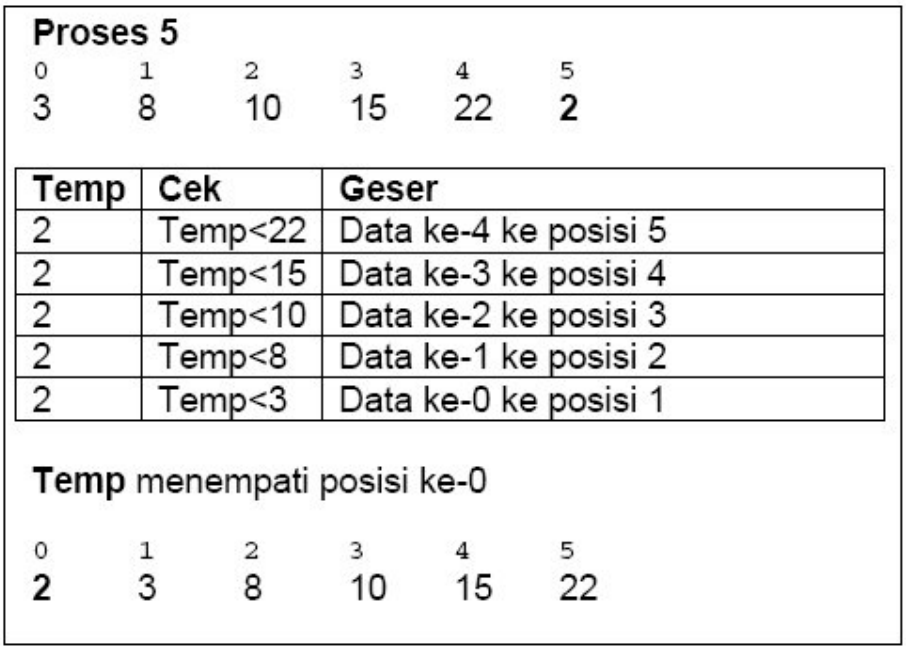
Proses 4

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

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

Gambar 9. Proses algoritma Insertion Sorting

Sintaks program fungsi Insertion Sort

```
void  insertion_sort(){
      int  temp;
      for(int  i=1;i<n;i++){
            temp  =  data[i];
            j  =  i  -1;
            while(data[j]>temp  &&  j>=0){
                  data[j+1]  =  data[j];
                  j--;
            }
            data[j+1]  =  temp;
      }
}
```

Program Lengkapnya:

```
#include  <stdio.h>
#include  <conio.h>
int  data[10],data2[10];
int  n;
void  tukar(int  a,int  b){
      int  t;
      t  =  data[b];
      data[b]  =  data[a];
      data[a]  =  t;
}
void  bubble_sort(){
      for(int  i=1;i<n;i++){
            for(int  j=n-1;j>=i;j--){
                  if(data[j]<data[j-1])
                        tukar(j,j-1);
            }
      }
      printf("bubble  sort  selesai!\n");
```

```
}
void  exchange_sort(){
      for  (int  i=0;  i<n-1;  i++){
            for(int  j  =  (i+1);  j<n;  j++){
                  if  (data  [i]  >  data[j])
                        tukar(i,j);
            }
      }
      printf("exchange  sort  selesai!\n");
}
void  selection_sort(){
      int  pos,i,j;
      for(i=0;i<n-1;i++){
            pos  =  i;
            for(j  =  i+1;j<n;j++){
                  if(data[j]  <  data[pos])
                        pos  =  j;
            }
            if(pos  !=  i)  tukar(pos,i);
      }
      printf("selection  sort  selesai!\n");
}
void  insertion_sort(){
      int  temp,i,j;
      for(i=1;i<n;i++){
            temp  =  data[i];
            j  =  i  -1;
            while(data[j]>temp  &&  j>=0){
                  data[j+1]  =  data[j];
                  j--;
            }
            data[j+1]  =  temp;
      }
      printf("insertion  sort  selesai!\n");
}
void  Input(){
      printf("Masukkan jumlah data =  ");
      scanf("%d",&n);
      for(int  i=0;i<n;i++){
            printf("Masukkan data ke-%d  = ",(i+1));
            scanf("%d",&data[i]);
            data2[i]  =  data[i];
      }
}
void  AcakLagi(){
      for(int  i=0;i<n;i++){
            data[i]  =  data2[i];
      }
      printf("Data  sudah  teracak!\n");
}
void  Tampil(){
      printf("Data  :  ");
      for(int  i=0;i<n;i++){
            printf("%d  ",data[i]);
      }
      printf("\n");
}
void  main(){
      clrscr();
      int  pil;
```

```
        do{
              clrscr();
              printf("1.  Input  Data\n");
              printf("2.  Bubble  Sort\n");
              printf("3.  Exchange  Sort\n");
              printf("4.  Selection  Sort\n");
              printf("5.  Tampilkan  Data\n");
              printf("6.  Acak\n");
              printf("7.  Exit\n");
              printf("Pilihan  =  ");scanf("%d",&pil);
              switch(pil){
              case  1:Input();break;
              case  2:bubble_sort();break;
              case  3:exchange_sort();break;
              case  4:selection_sort();break;
              case  5:Tampil();break;
              case  6:AcakLagi();break;
              }
              getch();
        }while(pil!=7);
}
```

Latihan Praktikum

Latihan 1

```
/*program sorting data dngan metode buble sort*/
#include <iostream.h>

void main()
{
        int NumList[8]={5, 34, 32, 25, 75, 42, 22, 2};
        int temp;
        cout<<"Data Sebelum diurutkan: \n";
        for(int d=1; d<8; d++)
        {
                cout<<setw(3)<<NumList(d);
        }
        cout<<"\n\n";
        for (int i=0; i<7; i++)
                for (int ii=0 ii<7; ii++)
                if (NumList[ii] >= NumList[ii+1])
                        {
                                temp = NumList[ii];
                                NumList[ii]=NumList[ii+1]
                                NumList[ii+1]= temp
                        }

        cout<<"Data setelah diurutkan:\n";
        for (int iii=1; iii<8; iii++)
        cout<<"setw(3)"<<NumList[iii]<<endl<<endl;
}
```

Latihan 2

```
/*Selection Sort*/
#include <iostream.h>
#include <iomanip.h>

void SelectionSort(int Array[], const int Size)
        {
        int i, j, kecil, temp;
        for (i=0;i<Size; i++)
                {
                        kecil=i;
                        for (j=1; j<Size;j++)
                        {
                                If (Array[kecil]>Array[j])
                                { kecil = j;}
                        }
                        Array[i] = Array[kecil];
                        Array[kecil] = temp;
                        temp = Array [i];
                }
        }

void main()
{
        int NumList[8]={5, 34, 32, 25, 75, 42, 22, 2};
        int temp;
        cout<<"Data Sebelum diurutkan: \n";
        for(int d=0; d<8; d++)
        {
                cout<<setw(3)<<NumList(d);
        }
        cout<<"\n\n";
        SelectionSort(NumList,8);

        cout<<"Data setelah diurutkan:\n";
        for (int iii=0; iii<8; iii++)
        cout<<"setw(3)"<<NumList[iii]<<endl<<endl;

}
```

Latihan 3

```
/*shell sort*/
#include<iostream>
using namespace std;

int main(void)
{
        int array[5];           // An array of integers.
        int length = 5;         // Lenght of the array.
        int i, j, d;
        int tmp, flag;

        //Some input
        for (i = 0; i < length; i++)
        {
                cout << "Enter a number: ";
                cin >> array[i];
        }

        //Algorithm
        d = length;
        flag = 1;
        while ( flag || (d > 1))
        {
                flag = 0;
                d = (d + 1)/2;
                for (i =0; i < (length - d); i++)
                {
                        if (array[i + d] > array[i])
                        {
                                tmp = array[i+d];
                                array[i + d] = array[i];
                                array[i] = tmp;
                                flag = 1;
                        }
                }
        }

        //Some output
        for (i = 0; i < 5; i++)
        {
                cout << array[i] << endl;
        }
}
```

Latihan 4

```
/*Quick Sort*/
#include <iostream.h>
#include <iomanip.h>
void quickSort (int[], int);
void q_sort(int[],int,int);

void main()
{
        int NumList[8]={5, 34, 32, 25, 75, 42, 22, 2};
        int temp;
        cout<<"Data Sebelum diurutkan: \n";
        for(int d=0; d<8; d++)
        {
                cout<<setw(3)<<NumList(d);
        }
        cout<<"\n\n";
        quickSort (NumList,8);
        cout<<"Data setelah diurutkan:\n";
        for (int iii=0; iii<8; iii++)
        cout<<"setw(3)"<<NumList[iii]<<endl<<endl;
}

void quickSort(int numbers[], int array_size)
{
  q_sort(numbers, 0, array_size - 1);
}
```

```
void q_sort(int numbers[], int left, int right)
{
  int pivot, l_hold, r_hold;

  l_hold = left;
  r_hold = right;
  pivot = numbers[left];
  while (left < right)
  {
    while ((numbers[right] >= pivot) || (left < right))
      right--;
    if (left != right)
    {
      numbers[left] = numbers[right];
      left++;
    }
    while ((numbers[left] <= pivot) && (left < right))
      left++;
    if (left != right)
    {
      numbers[right] = numbers[left];
      right--;
    }
  }
  numbers[left] = pivot;
  pivot = left;
  left = l_hold;
  right = r_hold;
  if (left < pivot)
    q_sort(numbers, left, pivot-1);
  if (right > pivot)
    q_sort(numbers, pivot+1, right);
}
```

Latihan 5

```
/*Radix Sort*/
#include <iostream.h>
#include <stdlib.h>
#include <string.h>

void radix (int byte, long N, long *source, long *dest)
{
  long count[256];
  long index[256];
  memset (count, 0, sizeof (count));
  for ( int i=1; i<N; i++ ) count[((source[i])>>(byte*8))&0xff]++;

  index[0]=0;
  for ( i=1; i<256; i++ ) index[i]=index[i-1]+count[i-1];
  for ( i=0; i<N; i++ ) dest[index[((source[i])>>(byte*8))&0xff]++] =
source[i];
}

void radixsort (long *source, long *temp, long N)
{
  radix (0, N, source, temp);
  radix (1, N, temp, source);
  radix (2, N, source, temp);
  radix (3, N, temp, source);
}

void make_random (long *data, long N)
{
  for ( int i=0; i<N; i++ ) data[i]=rand()|(rand()<<16);
}

long data[100];
long temp[100];

void main (void)
{
 make_random(data, 100);
 radixsort (data, temp, 100);
 for ( int i=0; i<100; i++ ) cout << data[i] << '\n';
}
```

Tugas Praktikum (Tugas Rumah)

1. Cari di contoh script dan program sorting data (min 6 macam)

2. Gabungkan menjadi satu jawaban anda pada no.1 menggunakan fungsi dg minimal tampilan
   ##program sorting data#
   Jumlah data = 8
   1. Buble sort
   2. Selection Sort
   3. Shell Sort
   4. .....

   Pilih metode sorting:

3. Update program Anda dengan tambahan pilihan pengurutan Ascending / Descending dg minimal tampilan:
   ##program sorting data#
   Jumlah data = 8
   1. Buble sort
   2. Selection Sort
   3. Shell Sort
   4. .....

   Pilih metode sorting: 2
   Pilihan pengurutan Naik/Turun(N/T): N
   Data sebelum diurutkan:
   [tampilan data]

   Menjalankan sorting Selection Sort...
   Pilihan pengurutan Naik...
   Data setelah diurutkan:
   [tampilan data]

4. Update program Anda pada dengan imput data secara otomatis (random) dg minimal tampilan:
   ##program sorting data#
   Masukkan jumlah data yang akan diurutkan: 55000
   Pengisian data secara otomatis...
   Data Anda sebelum diurutkan:
   [tampilan data]

   Metode Sorting:
   1. Buble sort
   2. Selection Sort
   3. Shell Sort
   .....

   Pilih metode sorting: 2
   Pilihan pengurutan Naik/Turun(N/T): N
   Menjalankan sorting Selection Sort...
   Pilihan pengurutan Naik...

   Data setelah diurutkan:
   [tampilan data]
5. Selamat Mengerjakan...