BAB 2

SORTING (PENGURUTAN)

1. Tujuan

Setelah mempelajari modul ini, mahasiswa diharapkan:

a. Mampu menjelaskan mengenai algoritma Sorting
b. Mampu membat dan mendeklarasikan struktural algoritma Sorting
c. Mampu menerapkan dan mengimplementasikan algoritma Sorting

2. Dasar Teori

Pengurutan data dalam struktur data sangat penting terutama untuk data yang bertipe 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	3 9 6 21 10 1 13
Ascending	1 3 6 9 10 13 21
Descending	21 13 10 9 6 3 1

Deklarasi Array Sorting

Mendeklarasikan array secara global:

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

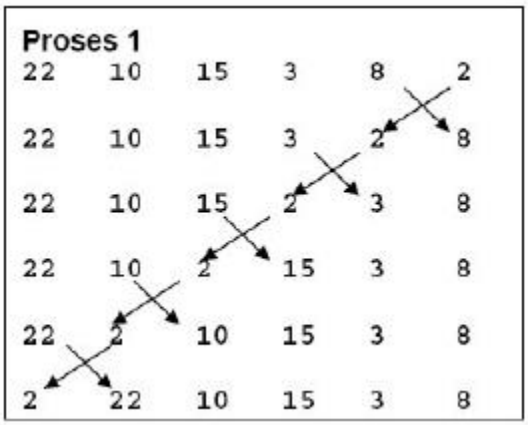
Fungsi Tukar 2 Buah Data:

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

BUBBLE SORT

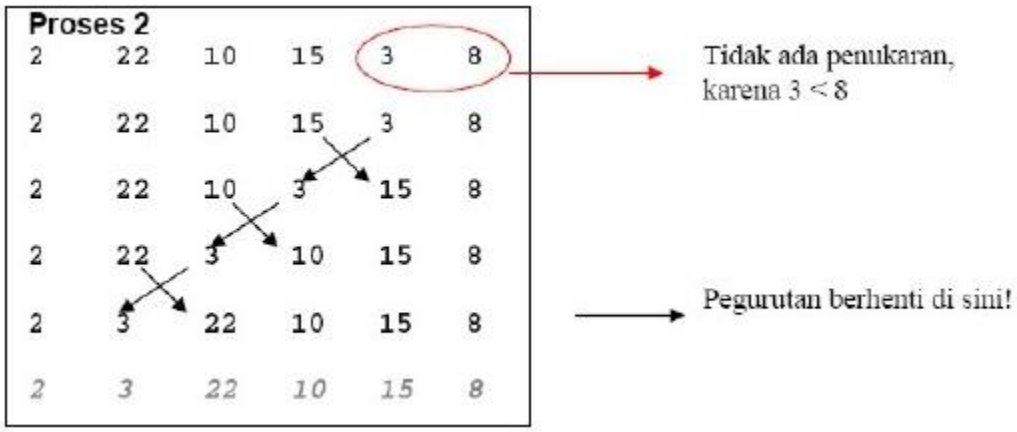
Merupakan metode sorting termudah, diberi nama "Bubble" karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda. *Bubble Sort* mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya. 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.



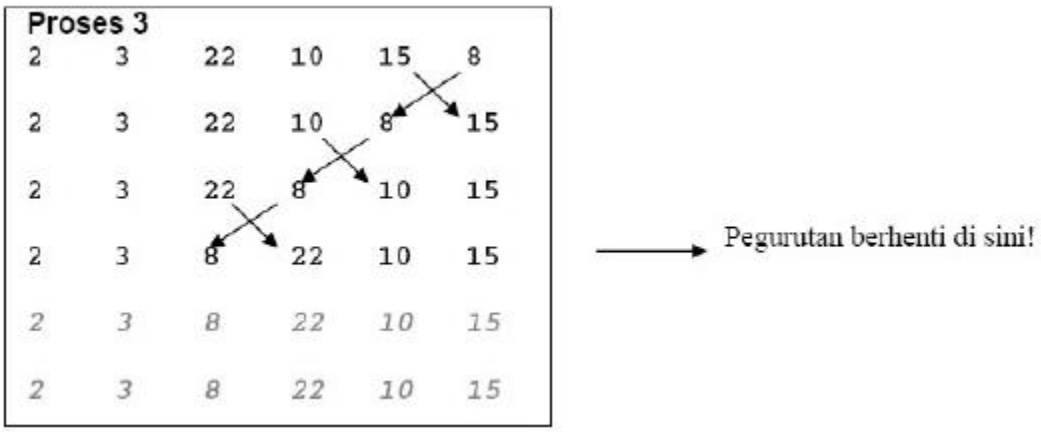
Gambar 1. Proses ke-1 Algoritma Bubble Sort

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

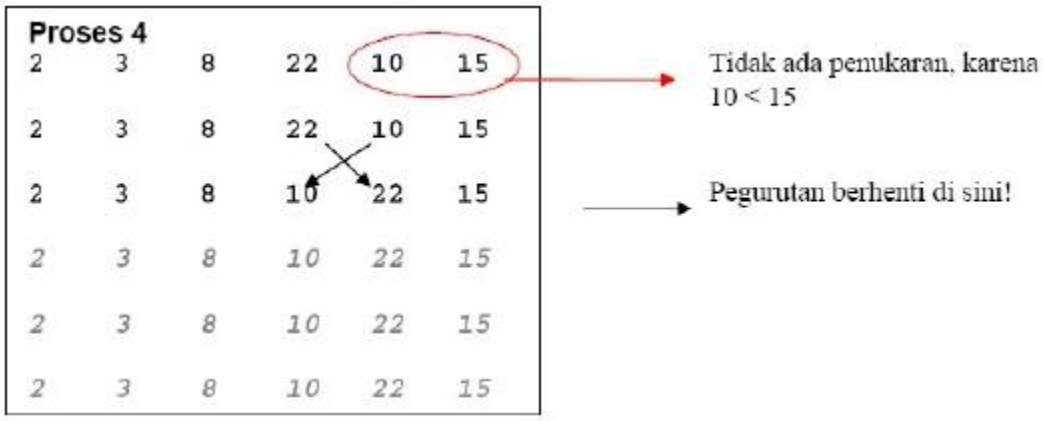


Gambar 2. Proses ke-2 Algoritma Bubble Sorting

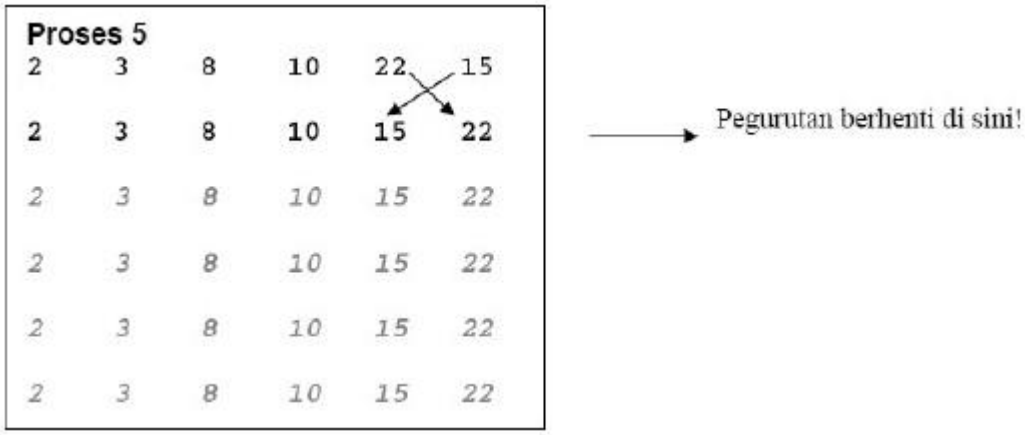
Pada proses ke-2, pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasting sudah paling kecil.



Gambar 3. Proses ke-3 Algoritma Bubble Sorting



Gambar 4. Proses ke-4 Algoritma Bubble Sorting



Gambar 5. Proses ke-5 Algortima Bubble Sorting

Sintaks program fungsi Bubble Sort

```
void bubble sort () {
      for (int i = 1; i<n; i++) {
            for (int j = n-1; j>=1; j--) {
                  if (data [j] < data [j-1])
                  tukar (j, j-1); // ascending
      }
}
```

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

```
if (data [j] < data [j-1] )
```

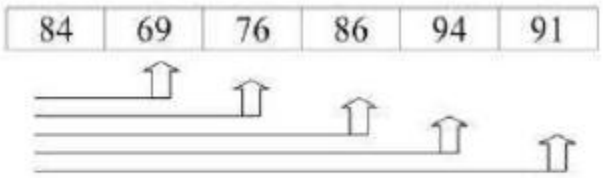
Menjadi:

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

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

EXCHANGE SORT

Sangat mirip dengan Bubble Sort, dan banyak yang mengatakan Bubble Sort sama dengan Exchange Sort. Perbedaan 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)

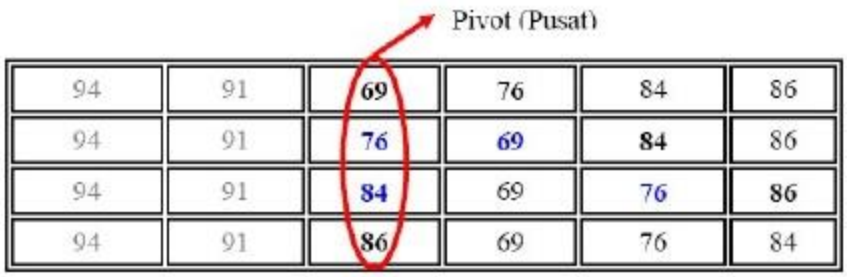
84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Proses 2

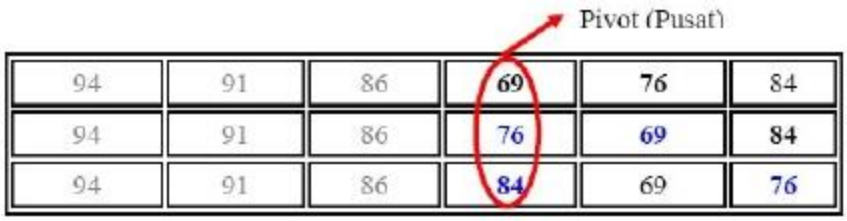
Pivot (Pusat)

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

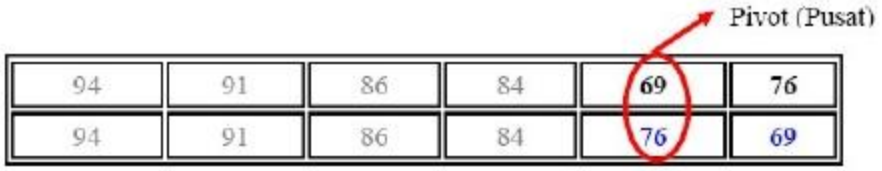
Proses 3



Proses 4



Proses 5



Sintaks program fungsi Exchange Sorting

```
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 pembandingan 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

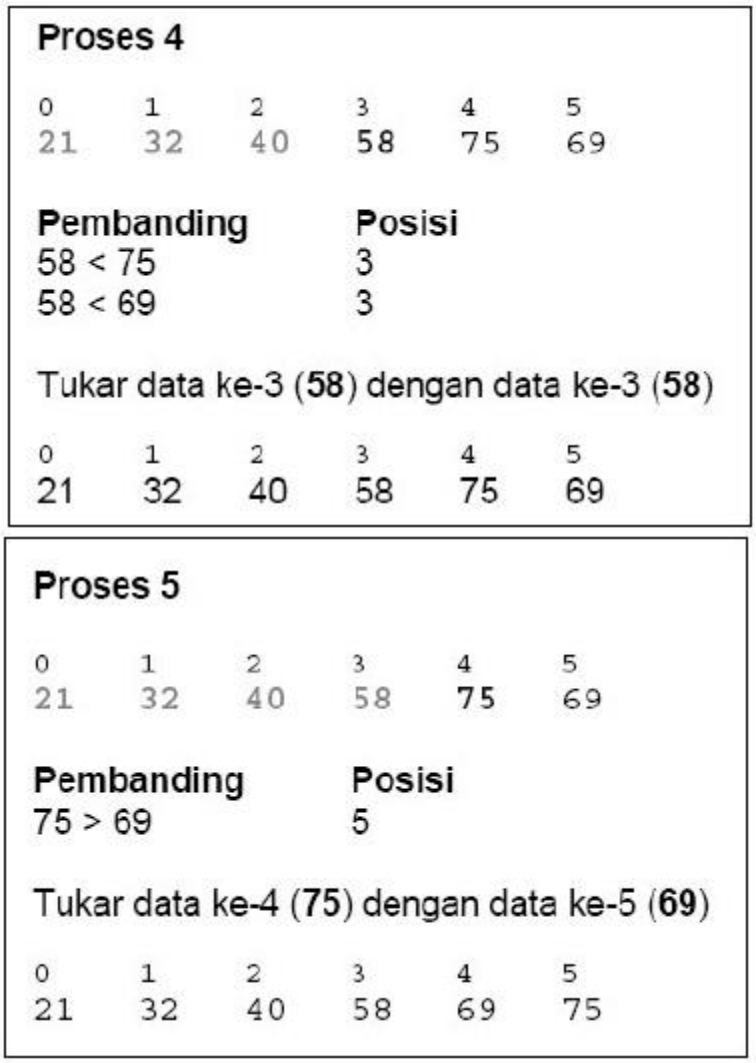
Proses 3

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

Pembanding	Posisi
69 > 58 (**tukar idx**)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (**69**) dengan data ke-5 (**40**)

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



Gambar 7. Proses Algoritma Selection Sorting

Sintaks program fungsi Selection Sorting

```
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);
        }
}
```

INSERT 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 (di-*insert*) 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: **(PERCOBAAN LATIHAN)**

```
#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++) {
                print (“%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”);
        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);
}
```

3. Latihan Praktikum

Latihan 1

Berikan algoritma dan penjelasan dari syntaks program di bawah ini!

```
/* Bubble Sorting */
#include <iostream>
#include <iomanip>

using namespace std;
int main()
{
    int NumList [10]={12,29,56,4,31,9,17,19,48,3};
    int temp;
    cout<<"Data Angka Sebelum diurutkan : \n";
    for (int d=0; d<10; d++)
        {
            cout<<setw(3)<<NumList[d];
    }
    cout<<"\n\n";
    for (int a=0; a<10; a++)
            for (int b=0; b<10; b++)
                if (NumList [b]>= NumList [b+1])
                {
                    temp = NumList[b];
                    NumList[b]= NumList [b+1];

                    NumList[b+1]= temp;
                }
                cout<<"Data setelah diurutkan:\n";
                for (int c=0; c<10; c++)
                cout<<setw(3)<<NumList[c]<<"  ";
                cout<<endl;
}
```

Latihan 2

Berikan algoritma dan penjelasan dari syntaks program di bawah ini!

```
/*Selection Sorting */
#include <iostream>
#include <iomanip>
using namespace std;

void SelectionSort (int Array[], const int Size)
{
     int i,j,small, temp;
     for (i=0;i<Size;i++)
     {
          small=i;
          for (j=0;j<Size;j++)
          {
               if (Array[j]>Array[small])//Pembanding
               {

                    small = j;
                    temp = Array[i];
          Array[i] = Array[small];
          Array[small]=temp;
               }

          }
     }
}

int main()
{
     int NumList [10]={12,29,56,4,31,9,17,19,48,3};
     int temp;
     cout<<"Data sebelum diurutkan: \n";
     for(int d=0; d<10; d++)
     {
          cout<<setw(3)<<NumList[d];
     }
     cout<<"\n\n";
     SelectionSort(NumList,10);

cout<<"Data setelah diurutkan\n";
     for (int a=0; a<10; a++)
          cout<<setw(3)<<NumList[a]<<endl<<endl;
}
```

Latihan 3

Berikan algoritma dan penjelasan dari syntaks program di bawah ini!

```
/* Shell Shorting */
#include <iostream>
using namespace std;

int main()
{
    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];
    }

//Algotithm
    d=length;
    flag=1;

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

Latihan 4

Berikan algoritma dan penjelasan dari syntaks program di bawah ini!

```
/*Quick Sorting*/
#include <iostream>
#include <iomanip>
using namespace std;

void quickSort (int[],int);
void q_sort (int[],int,int);

int main()
{
    int NumList[10]={12,29,56,4,31,9,17,19,48,3};
    int temp;
    cout<<"Data sebelum diurutkan:\n";
    for (int d=0;d<10;d++)
    {
        cout<<setw(3)<<NumList[d];
    }
    cout<<"\n\n";
    quickSort (NumList,10);
    cout<<"Data setelah diurutkan:\n";
    for (int iii=0; iii<10; 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

Berikan algoritma dan penjelasan dari syntaks program di bawah ini!

```
/*Radix Sorting*/
#include <iostream>
#include <stdlib.h>
#include <string.h>
using namespace std;

void radix (int byte, long N, long *source, long *dest)
{
    long count[256];
    long index[256];
    int i;
    memset (count, 0, sizeof(count));
    for (i=0; 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];
int main (void)
{
    make_random(data,100);
    radixsort (data, temp, 100);
    for (int i=0; i<100; i++) cout<<data[i]<<'\n';
}
```

4. Tugas Rumah

Buatlah sebuah program untuk mengurutkan sepasang data yang dimasukkan berdasarkan abjad/huruf.

Contoh

(Input) ;

Masukan huruf ke-1 : w

Masukan angka ke-1 : 7

Masukan huruf ke-2 : d

Masukan angka ke-2 :2

Masukan huruf ke-3 : b

Masukan angka ke-3 :0

Masukan huruf ke-4 : a

Masukan angka ke-4 :8

Masukan huruf ke-5 : z

Masukan angka ke-5 :4

(Output)

Data Sebelum Diurutkan :

w	d	b	a	z
7	2	0	8	4

Urutan Berdasarkan Huruf :

a	b	d	w	z
8	0	2	7	4

Keterangan: Data yang dihasilkan setelah diurutkan TETAP BERPASANG-PASANGAN.

... SELAMAT MENGERJAKAN ...