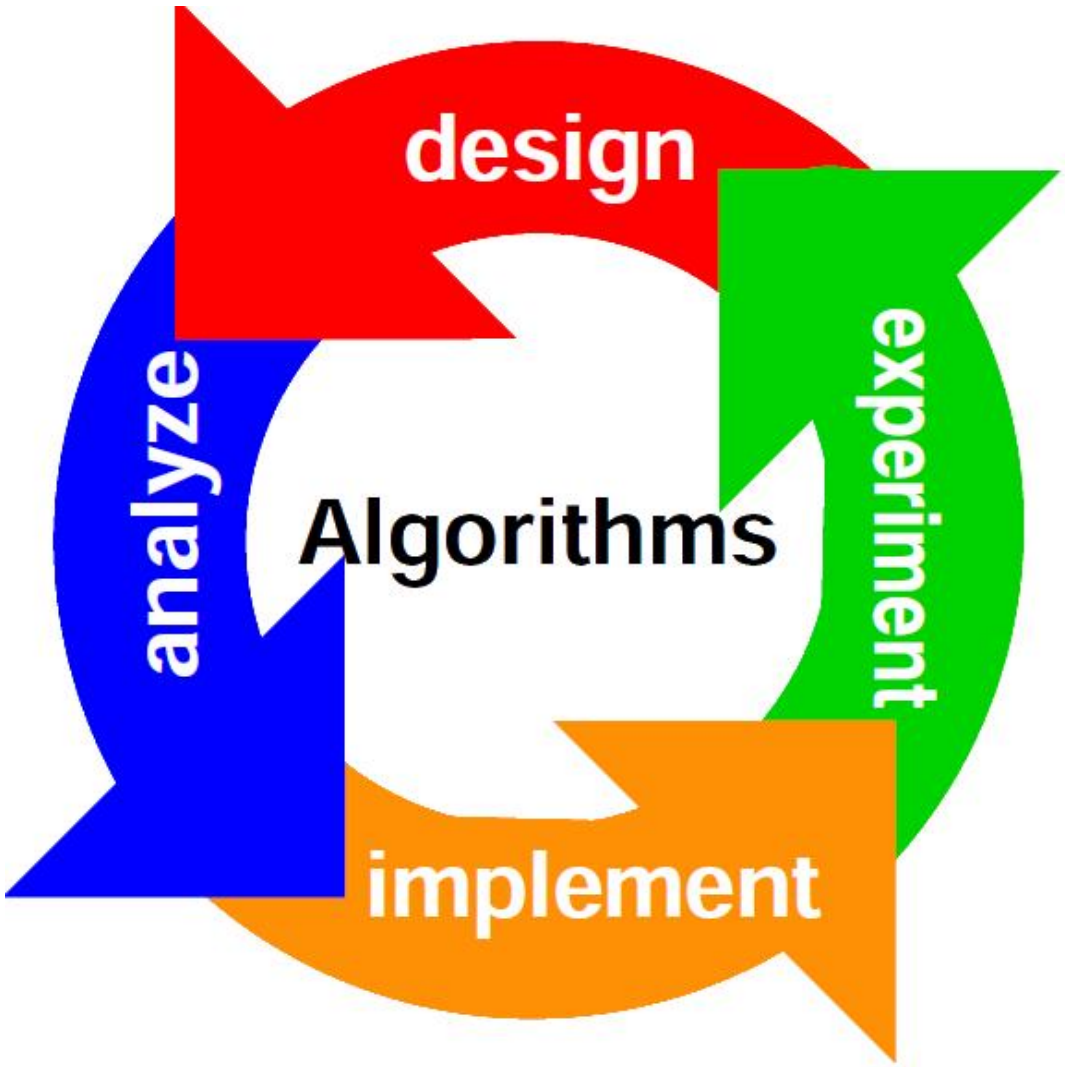
ALGORITMA DAN PEMROGRAMAN 2



TEKNIK PERANGKAT LUNAK
FT - UNPAM

3.PENGURUTAN (SORTING) lanjutan

III. METODE PENGURUTAN SISIPAN (INSERTION SORT)

Metode ini melakukan pengurutan dengan cara menyisipkan elemen array pada posisi yang tepat. Pencarian posisi yang tepat dilakukan dengan menyisir larik. Selama penyisiran, dilakukan pergeseran elemen array.

Algoritma pengurutan sisipan untuk pengurutan menaik

Untuk mendapatkan array yang terurut menaik , algoritma pengurutan sisipan secara garis besar ditulis sbb:

Jumlah elemen : N Indeks tetendah : 0 Indeks tertinggi : n = N - 1	Untuk setiap tahap i = 1, 3, …, n ; lakukan: 1) x ← L[i] 2) sisipkan x pada tempat yang sesuai antara L[0]…L[i] Jumlah penyisipan = N - 1

L =	15	10	7	22	17	5	22
i =	0	1	..	..	..	..	n

Rincian setiap tahap adalah sebagai berikut:

TAHAP I:

Mulai dari L[1] : Simpan nilai L[1] ke variabel x

- Geser masing-masing satu langkah ke kanan semua nilai yang ada disebelah kiri L[1] satu persatu apabila nilai tersebut lebih besar dari x
- Setelah itu sisipkan x dibekas tempat nilai yang terakhir digeser

TAHAP II:

Dilanjutkan ke L[2] : Simpan nilai L[2] ke variabel x

- Geser masing-masing satu langkah ke kanan semua nilai yang ada disebelah kiri L[2] satu persatu apabila nilai tersebut lebih besar dari x
- Setelah itu sisipkan x dibekas tempat nilai yang terakhir digeser..

Demikian seterusnya untuk elemen berikutnya, dan terakhir L[6]. Sehingga untuk jumlah elemen = 7 maka ada 6 tahap proses.

Algoritma penyisipan menaik dapat ditulis:

```
procedure Insertion(input/output L : LarikInt, input n : integer)
DEKLARASI
  i       : integer    {Pencacah tahap}
  j       : integer    {Pencacah untuk penelusuran larik}
  x       : integer    {Peubah bantu}
  ketemu : boolean     {Untuk menyatakan posisi penyisipan ditemukan}
DESKRIPSI
      for i ← 1 to n do
          x ← L[i]
          j ← i - 1
          ketemu = false
      while((j ≥ 0) and (not ketemu)) do
          if x < L[j] then
              L[j + 1] ← L[j] {geser}
              j ← j - 1
          else
              ketemu ← true
          endif
      endwhile
      L[j + 1] ← x
  endfor
```

Metode ini memerlukan banyak operasi pergeseran untuk tiap tahapnya sehingga kurang bagus untuk jumlah data yang banyak.

Program C++ untuk pengurutan sisipan menaik:

```
/*Mengurutkan data dengan metode Penyisipan*/
#include<iostream.h>
#define N 10 /*Jumlah data*/
void InsertionSort(int data[],int n); /*prototipe fungsi*/
void main(void)
{
```

```
void InsertionSort(int array1[],int n)
{
        int i,j,x;
        bool ketemu;
        for(i=1;i<=n;i++)
        {
```

```
        int i;
        int n=N-1;
        int data[]={20,10,32,100,60,12,70,25,45,65};
        cout<<"Sebelum diurutkan :"<<endl;
        for(i=0;i<=n;i++)
                cout<<data[i]<<" ";
        cout<<endl;
        cout<<"_____"
<<endl;
        InsertionSort(data,n);
        cout<<"Setelah diurutkan"<<endl;
        for(i=0;i<=n;i++)
                cout<<data[i]<<" ";
        cout<<endl;
}
```

```
                x=array1[i];
                j=i-1;

        ketemu=false;
                while((j>=0)
&& (!ketemu))
                {

        if(x<array1[j])
                        {

        array1[j+1]=array1
[j];

        j=j-1;
                        }
                        else

        ketemu=true;
                }

        array1[j+1]=x;
        }
}
```

Contoh :

Urutkan secara menaik elemen dari array sbb:

A :	15	10	7	22	17	5	12
Index :	0	1	2	3	4	5	6

⇓ Mulai dari index 1

TAHAP I :	15	10	7	22	17	5	12

X = A[1] = 10

A[0] > X, geser A[0] kekanan, proses selesai karena sudah sampai index 0

15	15	7	22	17	5	12

Sisipkan X menggantikan A[0] (index terakhir yg digeser)

10	15	7	22	17	5	12

⇓ Mulai dari index 2

TAHAP II :

10	15	7	22	17	5	12

X = A[2] = 7

A[1] > X, geser A[1] kekanan

10	15	15	22	17	5	12

A[0] > X, A[0] geser kekanan, proses selesai karena sudah sampai index 0

10	10	15	22	17	5	12

Sisipkan X menggantikan A[0] (index terakhir yg digeser)

7	10	15	22	17	5	12

⇓ Mulai dari index 3

TAHAP III :

7	10	15	22	17	5	12

X = A[3] = 22

A[2] !> X, proses selesai, sisipkan X ke A[3]

7	10	15	22	17	5	12

⇓ Mulai dari index 4

TAHAP IV :

7	10	15	22	17	5	12

X = A[4] = 17

A[3] > X, geser A[3] kekanan

7	10	15	22	22	5	12

A[2] ! > X, proses selesai , sisipka X ke A[3](terakhir digeser)

7	10	15	17	22	5	12

⇓ Mulai dari index 5

TAHAP V :

7	10	15	17	22	5	12

X = A[5] = 5

A[4] > X, geser A[4] kekanan

7	10	15	17	22	22	12

A[3] > X, geser A[3] kekanan

7	10	15	17	17	22	12

A[2] > X, geser A[2] kekanan

7	10	15	15	17	22	12

A[1] > X, geser A[1] kekanan

7	10	10	15	17	22	12

A[0] > X, geser A[0] kekanan, proses selesai karena sudah sampai index 0

7	7	10	15	17	22	12

sisipka X ke A[0](terakhir digeser)

5	7	10	15	17	22	12

⇓ Mulai dari index 6

TAHAP VI :

5	7	10	15	17	22	12

X = A[6] = 12

A[5] > X, geser A[5] kekanan

5	7	10	15	17	22	22

A[4] > X, geser A[4] kekanan

5	7	10	15	17	17	22

A[3] > X, geser A[3] kekanan

5	7	10	15	15	17	22

A[2] !> X, proses selesai , sisipka X ke A[3](terakhir digeser)

5	7	10	12	15	17	22

IV. METODE PENGURUTAN SHELL

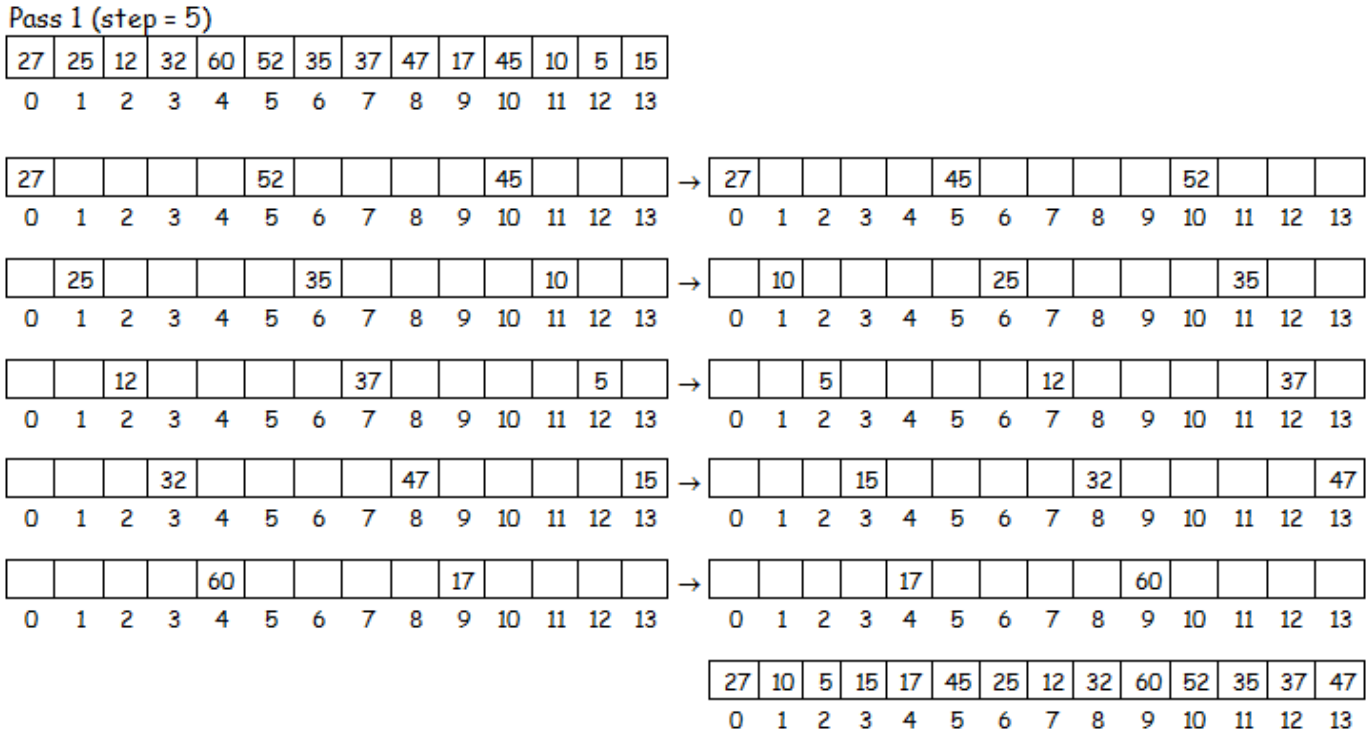
Diberi nama sesuai dengan penemunya yaitu Donald Shell tahun 1959. Metode ini merupakan perbaikan terhadap metode pengurutan sisipan. Insertion sort mempunyai keunggulan bila data yang diurutkan sudah hampir urut. Tetapi untuk data yang urutannya terbalik atau banyak data yang berada didaerah berlawanan maka akan banyak sekali proses penggeseran.

Untuk mengurangi pergeseran terlalu jauh maka larik diurutkan setiap k elemen dengan menggunakan metode pengurutan sisip(k dinamakan step atau increment), misal k=5. Selanjutnya digunakan nilai step yang lebih kecil, misal k=3, lalu diurut setiap 3 elemen. Begitu seterusnya sampai nilai k=1.

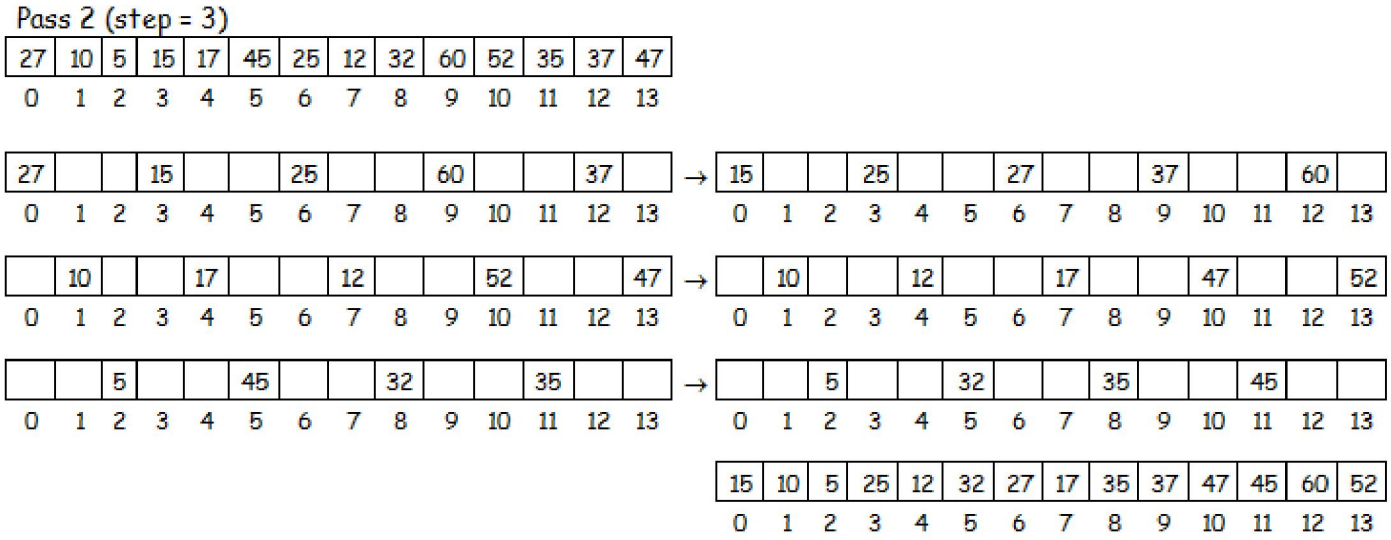
Untuk menjelaskan metode ini, tinjaulah sebuah data sbb:

A :	27	25	12	32	60	52	35	37	47	17	45	10	5	15
indek :	0	1	2	3	4	5	6	7	8	9	10	11	12	13

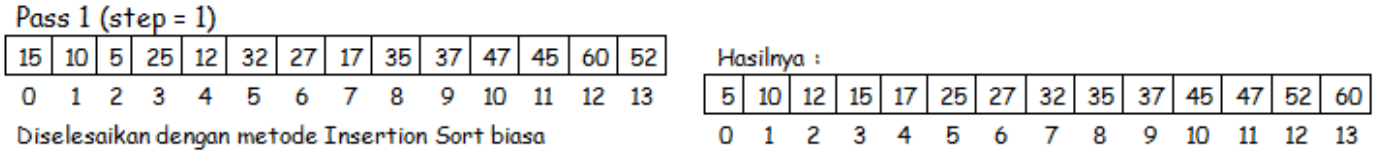
Pass 1 (step = 5)



Hasil Pass 1 adalah : 27, 10, 5, 15, 17, 45, 25, 12, 32, 60, 52, 35, 37, 47



Hasil pass 2 adalah : 15, 10, 5, 25, 12, 32, 27, 17, 35, 37, 47, 45, 60, 52
Terlihat bahwa hampir semua nilai kecil berada di bagian kiri, sehingga pada pass terakhir dapat dilakukan pengurutan metode sisipan untuk seluruh data dengan lebih efisien



Notasi Algoritmik :

```
procedure Insertion(input/output L : LarikInt, input n, start,
                    step : integer)
DEKLARASI
  i       : integer   {Pencacah tahap}
  j       : integer   {Pencacah untuk penelusuran larik}
  x       : integer   {Peubah bantu}
  ketemu  : boolean   {Untuk menyatakan posisi
                       penyisipan ditemukan}
```

```
procedure Shell(input/output L : LarikInt, input n : integer)
DEKLARASI
  step, start : integer
DESKRIPSI
  step ← 5
  while step >= 1 do
    for start ← 0 to step do
      Insertion(L, n, start, step)
    endfor
    step ← step - 2
  endwhile
```

```
DESKRIPSI
  i ← start + step
  while i ≤ n  do
      x ← L[i]
      j ← i - step
       ketemu = false
       while((j ≥ 0) and (not ketemu)) do
          if x < L[j] then
             L[j + step] ← L[j]   {geser}
              j ← j -  step
          else
              ketemu ← true
         endif
      endwhile
      L[j + step] ← x
  endwhile
```

Program C++ :

```
#include<iostream.h>
#define N 10 /*Jumlah data*/
void InsertionSort(int data[],int n,int start,int step);
void ShellSort(int data[],int n);
void main(void)
{
        int i, n=N-1;
        int data[]={20,10,32,100,60,12,70,25,45,65};
        cout<<"Sebelum diurutkan :"<<endl;
        for(i=0;i<=n;i++)cout<<data[i]<<" ";
        cout<<endl;
        cout<<"_____"
<<endl;
        ShellSort(data,n);
        cout<<"Setelah diurutkan"<<endl;
        for(i=0;i<=n;i++)cout<<data[i]<<" ";
        cout<<endl;
```

```
void InsertionSort(int
data[],int n, int start, int
step)
{
        int i,j,x;
        bool ketemu;
        i=start+step;
        while(i<=n)
        {
                x=data[i];
                j=i-step;

        ketemu=false;
                while((j>=0)
&& (!ketemu))
                {
```

```
}
void ShellSort(int data[],int n)
{
        int start,step;
        for(step=5;step>=1;step-=2)
        {
                for(start=0; start<=step;start++)
                        InsertionSort(data,n,start,step);
        }
}
```

```
        if(x<data[j])
                        {

        data[j+step]=data
[j];

        j=j-step;
                        }
                        else

        ketemu=true;
                }

        data[j+step]=x;
                i=i+step;
        }
}
```

TUGAS

1. Buatlah program C++ untuk mengurutkan data berikut secara menurun dengan menggunakan metode Sisipan:

A : 27 25 12 32 60 52 35 37 47 17 45 10 5 15

2. Buatlah program C++ untuk mengurutkan data berikut secara menurun dengan menggunakan metode Shell:

A : 27 25 12 32 60 52 35 37 47 17 45 10 5 15