HAND OUT

ALGORITHM AND PROGRAMMING II

Wilis Kaswidjanti, S.Si., M.Kom.

Department of Informatics Engineering

Faculty of Industrial Engineering

Universitas Pembangunan Nasional Veteran Yogyakarta

MATRIX
(Two- Dimensional Array)

Algorithms and Programming II

Wilis Kaswidjanti

Informatika UPN "Veteran" Yk

ARRAY

- Array is a collection of data where each element wears the same name and the same type and each element accessed by distinguishing the associated array index .
- Array Declaration

  Array variables are declared by stating the type and variable name followed by the number of memory locations to be made .
- example :
  - int a[4]; → 1 dimensional array
  - int b[2][3]; → 2 dimentional array

Two-Dimentional Array

- Two-dimensional array is an array consisting of m pieces of rows and n columns. This may be as a matrix or table .
- Declaration array (Pseudocode) :

```
nama_var : array [rangeindeks1,rangeindeks2] of tipe
```

  - example : x : array[1..2,1..3] of integer
- Declaration array (C++):

```
tipe nama_var[ukuran1][ukuran2];
```

  - example : int x[2][3];

Two-Dimensional Array

- example :

int a[2][3] = {{11, 7, 4},{12, 3, 9}};

which will occupy the memory location with the following composition:

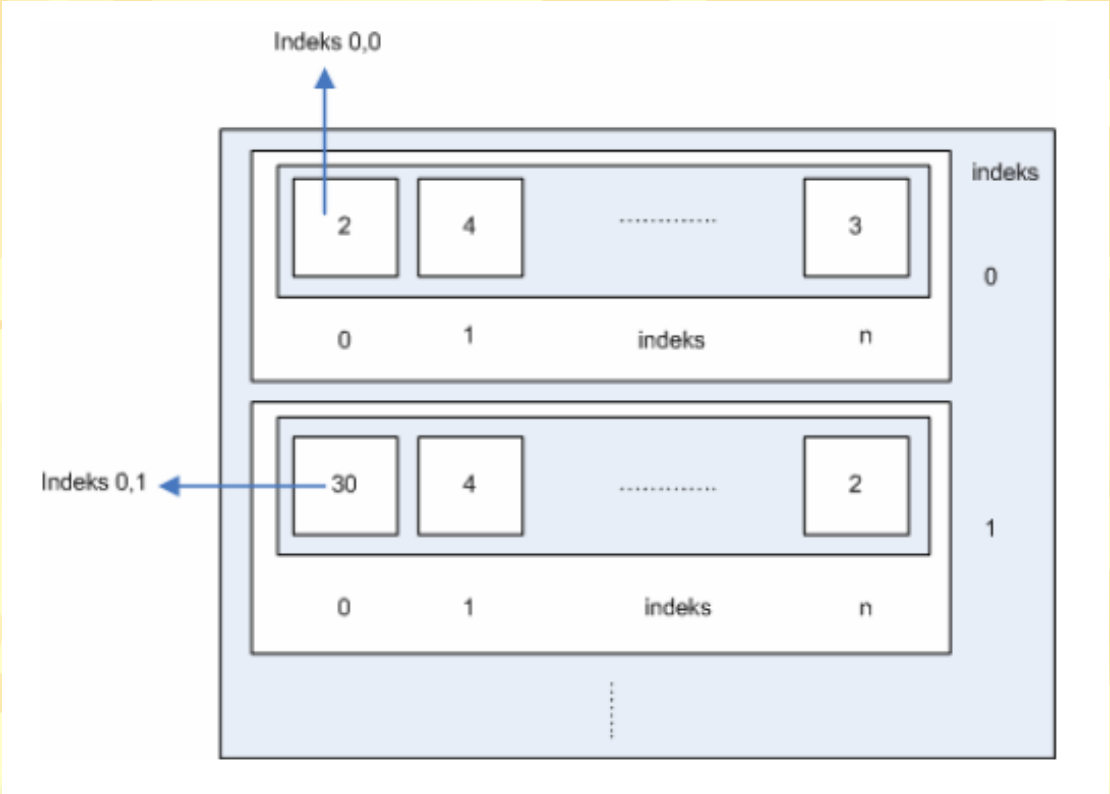
	0	1	2
0	11	7	4
1	12	3	9

and variable definitions for each of these elements are:

	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]

Matrix (Two-Dimensional Array)

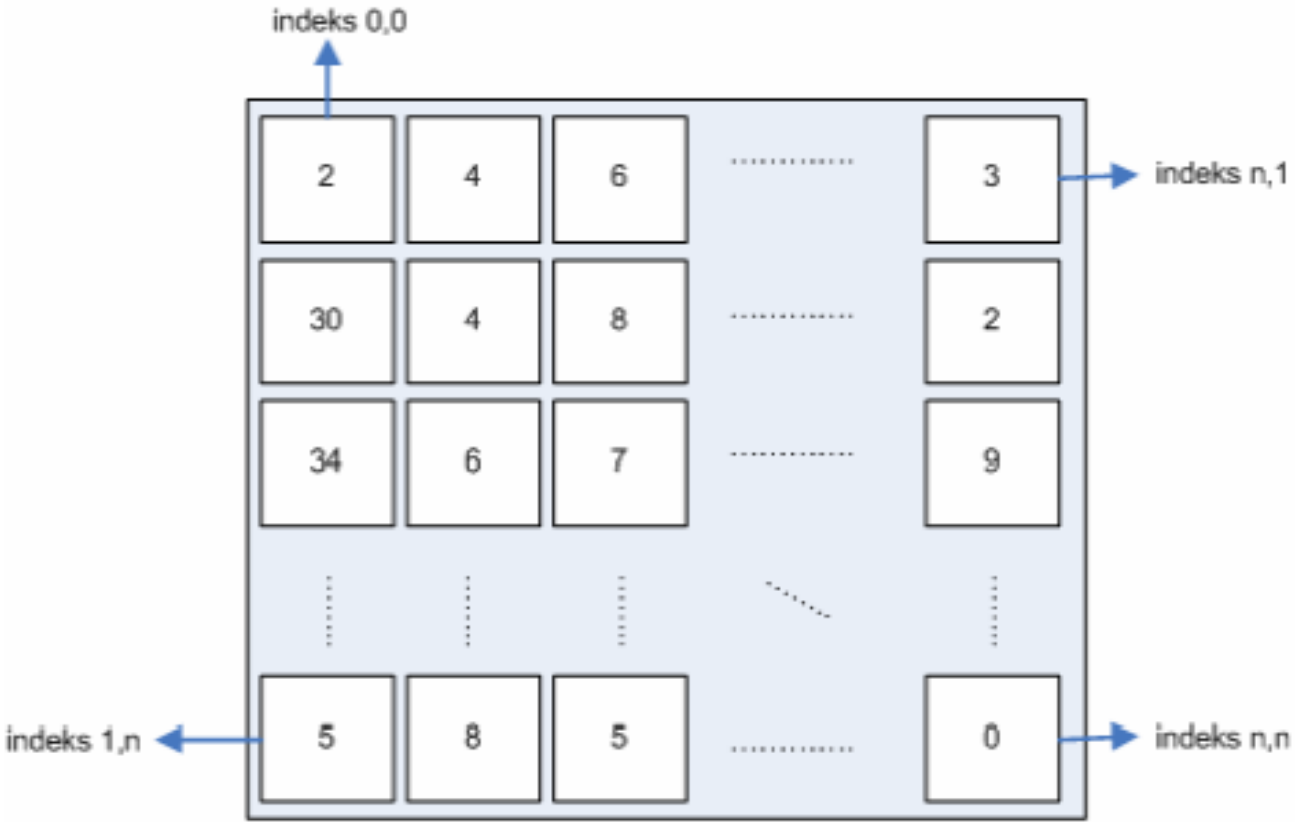
- Two-dimensional matrix or array is an array that has two or more columns with multiple lines, or two or more rows with multiple columns .
- Illustration:



Matrix (Two-Dimensional Array)

- Is actually a two-dimensional array is an array that is in the array.
- For example : there is a two-dimensional array of size 2 x 2 then in a first cell where there will be a one-dimensional array in it .
- So the picture is easier to represent a two-dimensional

Two-Dimensional Array Representation



Two-Dimensional Array Representation

Deklarasi array dua dimensi adalah sebagai berikut :

Bahasa Algoritmik	Bahasa Pascal	Bahasa C
nama_array : array [1…nBaris, 1…nKolom] of tipe_data	type nama_tipe_array = array[1..jumlah_baris, 1..jumlah_kolom] of tipe_data; var nama_array : nama_tipe_array;	tipe_data nama_array[jumlah_baris] [jumlah_kolom];
tabInt : array [1…20, 1…10] of integer	type tabel = array [1..20, 1..10] of integer; var tabInt : tabel;	int tabInt[20][10];

Two-Dimensional Array Representation

sedangkan cara mengaksesnya adalah sebagai berikut :

Keterangan	Bahasa Algoritmik	Bahasa Pascal	Bahasa C
mengisi nilai array dengan indeks tertentu	nama_array$_{baris, kolom}$ <- nilai	nama_array[baris, kolom] := nilai;	nama_array[baris][kolom] = nilai;
	tabInt$_{1, 1}$ <- 5	tabInt[1, 1] := 5;	tabInt[0][0] = 5;
mengakses nilai array dengan indeks tertentu	nama_array$_{baris, kolom}$	nama_array[baris, kolom];	nama_array[baris][kolom];
	tabInt$_{1, 1}$	tabInt[1, 1];	tabInt[0][0];

Process matrix (matrix filling)

- Charging can be done by using a matrix of repetition , Here is the algorithm to fill the matrix :

Algorithms filling matrix

Bahasa Manusia	Bahasa Algoritmik
Membuat matriks yang akan diisi	`matriks : array [1..4, 1..4] of integer`
Membuat kotak penghitung baris dan penghitung kolom untuk melakukan pengulangan pengisian matriks	`penghitung_baris : integer` `penghitung_kolom : integer`

Melakukan pengulangan per baris
Melakukan pengulangan per kolom
meminta masukan *user* /pemakai program untuk memasukkan elemen matriks
	`for penghitung_baris <- 1 to 4 do`
	`for penghitung_kolom <- to 4 do`
	`input (matriks`$_{penghitung\_baris,\ penghitung\_kolom}$`)`
	`{end for}`

Secara logika sebuah matriks diisi per baris dan setiap barisnya diisi per kolom	

Showing element matrix

- Featuring elements of the matrix can be performed by using repetition .
- Courtney: matrix must be filled in order to display its contents .
- The algorithm ...

Bahasa Manusia	Bahasa Algoritmik
Membuat matriks yang akan diisi	`matriks : array [1..4, 1..4] of integer`
Membuat kotak penghitung baris dan penghitung kolom untuk melakukan pengulangan pengisian matriks	`penghitung_baris : integer` `penghitung_kolom : integer`

Melakukan pengisian matriks
Melakukan pengulangan per baris
 	
Melakukan pengulangan per kolom
 	
meminta masukan *user* /pemakai program untuk memasukkan elemen matriks
 	
Secara logika sebuah matriks diisi per baris dan setiap barisnya diisi per kolom
	`for penghitung_baris <- 1 to 4 do`
	
	`for penghitung_kolom <- to 4 do`
	
	`input(matriks`$_{penghitung\_baris,\ penghitung\_kolom}$`)`
	
	`{end for}`
	
	`{end for}`

Melakukan penampilan matriks ke layar
 	
Melakukan pengulangan per baris
	`for penghitung_baris <- 1 to 4 do`
	
	`for penghitung_kolom <- to 4 do`

next...

Bahasa Manusia	Bahasa Algoritmik
Melakukan penampilan matriks ke layar	
Melakukan pengulangan per baris	`for penghitung_baris <- 1 to 4 do`
	`for penghitung_kolom <- to 4 do`
Melakukan pengulangan per kolom	
menampilkan isi matriks ke layar	`output(matriks`$_{\text{penghitung\_baris, penghitung\_kolom}}$`, "  ")`
menampilkan ganti baris	`{end for}`
	`output("\n")`
	`{end for}`

copy the matrix

- Copy the matrix can be performed by using repetition .
- Copy matrix means copying the contents of the matrix into another matrix has a dimension / the same size .

copy the matrix

Bahasa Manusia	Bahasa Algoritmik
Membuat matriks yang akan diisi dan matriks yang akan diisi dengan hasil salinan matriks yang telah diisi sebelumnya	matriks1 : array [1..4, 1..4] of integer matriks2 : array [1..4, 1..4] of integer
Membuat kotak penghitung baris dan penghitung kolom untuk melakukan pengulangan pengisian matriks	penghitung_baris : integer penghitung_kolom : integer

Melakukan pengulangan per baris
 	
Melakukan pengulangan per kolom
 	
meminta masukan *user* /pemakai program untuk memasukkan elemen matriks
	for penghitung_baris <- 1 to 4 do
	
	for penghitung_kolom <- to 4 do
	
	input(matriks1penghitung_baris, penghitung_kolom)
	
	{end for}
	
	{end for}

next...

Bahasa Manusia	Bahasa Algoritmik
Secara logika sebuah matriks diisi per baris dan setiap barisnya diisi per kolom	
Melakukan pengulangan per baris	for penghitung_baris <- 1 to 4 do
Melakukan pengulangan per kolom	for penghitung_kolom <- to 4 do
mengisi matriks2 dengan isi dari matriks1	matriks2penghitung_baris, penghitung_kolom <- matriks1penghitung_baris, penghitung_kolom
	{end for}
Secara logika sebuah matriks diisi per baris dan setiap barisnya diisi per kolom	{end for}

Sample Program :

```
#include<stdio.h>
void printArray(int a[][3]);
main()
{   int matrik1[2][3] = {{1, 2, 3}, {4, 5, 6}};
    int matrik2[2][3] = {{1, 2, 3}, {4, 5}};
    int matrik3[2][3] = {{1, 2}, {4} };
    printArray(matrik1);
    printArray(matrik2);
    printArray(matrik3);
}
void printArray(int a[][3])
{   int i, j;
    for(i=0; i<=1; i++)
    {  for(j=0; j<=2; j++)
          cout << a[i][j];
       cout << "\n";
    }
}
```

Output :

123
456
123
450
120
400

Matrix

- example :

  Unknown 2 pieces matrices A and B respectively berordo 3 x 2

A

$$\begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 4 & 6 \end{bmatrix}$$

B

$$\begin{bmatrix} 4 & 5 \\ 5 & 2 \\ 2 & 1 \end{bmatrix}$$

Create an algorithm / program to count the number of the second matrix .

How about

reduction matrix ?

and

Matrix multiplication ?

EXERCISE

1. Unknown matrix A and matrix B as follows :

$$A = \begin{bmatrix} 2 & 3 & 6 \\ 4 & 7 & 8 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 5 \\ 2 & 4 \\ 8 & 6 \end{bmatrix}$$

Make a program to compute

matrix C = matrix A * matrix B

referensi

- Hadi Sisyanto, S.Kom., Matriks (Array 2 Dimensi), Politeknik NSC Surabaya

RECORD/STRUCT and ARRAY OF RECORD

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

RECORD/STRUCT

- Record/Struct has a number of elements called fields.
- If all elements of the array must have the same data type , then each element in the record/struct can have different data types.

The general form Declaration Record/Struct

(algorithm) :

```
namaVar : record
    <  namaField1 : tipeField1 ,
       namaField2 : tipeField2 ,
       … …
       namaFieldm : tipeFieldm ,
    >
```

(C++) :

```
struct namaTipeStruct
    {
       tipeField1 namaField1;
       tipeField2 namaField2;
       … …
       tipeFieldm namaFieldm;
    } namaVar;
```

namaVar can be more than one

- Namavar can be separated from the declaration tiperecordnya , thus becoming:

```
typedef struct
    {
        tipeField1 namaField1;
        tipeField2 namaField2;
        ... ...
        tipeFieldm namaFieldm;
    } namaTipeStruct;

namaTipeStruct namaVar;
```

- To access the elements of record / structur done by :

```
namaVar.namaField
```

ARRAY OF RECORD

(algorithm) :

```
Type namaTipeRecord : record
   <  namaField1 : tipeField1 ,
      namaField2 : tipeField2 ,
      ... ...
      namaFieldm : tipeFieldm ,
   >
namaVar : array[rangeindex] of namaTipeRecord
```

(C++) :

```
typedef struct
    {
       tipeField1 namaField1;
       tipeField2 namaField2;
       ... ...
       tipeFieldm namaFieldm;
     } namaTipeStruct;
namaTipeStruct namaVar[ukuran];
```

To access the elements of record / structur is done by:

Algo: $namaVar_{index}$.namaField

C++ : namaVar[index].namaField

Example

(algorithm) :

```
Kamus
    Const Nmaks = 100;
    Type Mahasiswa : Record <  NIM : integer,
                               Nama : string,
                               IPK : real      >
    Mhs : Mahasiswa
    ArrayMhs : Array[1..Nmaks] of Mahasiswa
```

(C++) :

```
#define Nmaks 100
typedef struct
          {   int NIM;
              char Nama[25];
              float IPK;
          } Mahasiswa;
Mahasiswa Mhs;
Mahasiswa ArrayMhs[Nmaks];
```

Sample program :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
typedef struct
          {  int tanggal, bulan, tahun;
          } data_tanggal;
typedef struct
            {  char nama[30];
               data_tanggal tgl_lahir;
            } data_rekan;
data_rekan info_rekan;
```

```
main()
{  strcpy(info_rekan.nama,"Hanif");
   info_rekan.tgl_lahir.tanggal = 30;
   info_rekan.tgl_lahir.bulan = 4;
   info_rekan.tgl_lahir.tahun = 2002;
   clrscr();
   cout<<"Nama   : "<<info_rekan.nama;
   cout<<"\nTanggal lahir :";
   cout<<info_rekan.tgl_lahir.tanggal;
   cout<<"-"<<info_rekan.tgl_lahir.bulan;
   cout<<"-"<<info_rekan.tgl_lahir.tahun;
   getch();
}
```

How to declare a variation example program Structur :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
main()
{
      struct data_tanggal
       {  int tanggal, bulan, tahun; };
      struct data_rekan
       {  char nama[30];
          struct data_tanggal  tgl_lahir;
       };
  struct data_rekan info_rekan;
```

```
  strcpy(info_rekan.nama,"Hanif");
  info_rekan.tgl_lahir.tanggal = 30;
  info_rekan.tgl_lahir.bulan = 4;
  info_rekan.tgl_lahir.tahun = 2002;
  clrscr();
  cout<<"Nama   : "<<info_rekan.nama;
  cout<<"\nTanggal lahir :";
  cout<<info_rekan.tgl_lahir.tanggal;
  cout<<"-"<<info_rekan.tgl_lahir.bulan;
  cout<<"-"<<info_rekan.tgl_lahir.tahun;
  getch();
}
```

Exercise

1. Create a program for inputting and displaying 5 student data subjects and Programming Algorithm 2 with fields NoMhs, Name, Class, NilaiAngka and NilaiHuruf, provided NilaiHuruf not entered but comes from NilaiAngka . Range NilaiHuruf : 0<E<20; 20≤D<40; 40≤C<60; 60≤B<75; 75≤A<100

2. Create a program to input and display 2 student data .
   Example display :

Input Data

```
Banyaknya mahasiswa = 2

Mhs[0].NoMhs  = 123060200
Mhs[0].Nama   = Hanif
Mhs[0].BanyakMKA = 2
  Mhs[0].MKA[0].KodeMKA = 1
  Mhs[0].MKA[0].Kelas   = A
  Mhs[0].MKA[0].NilaiUTS= 95
  Mhs[0].MKA[1].KodeMKA = 4
  Mhs[0].MKA[1].Kelas   = C
  Mhs[0].MKA[1].NilaiUTS= 80

Mhs[1].NoMhs  = 123060201
Mhs[1].Nama   = Fathimah
Mhs[1].BanyakMKA     = 1
  Mhs[1].MKA[0].KodeMKA = 3
  Mhs[1].MKA[0].Kelas   = A
  Mhs[1].MKA[0].NilaiUTS= 90
```

Output Data

```
NoMhs   : 123060200
Nama    : Hanif

 Kode MKA    Kelas     NilaiUTS
    1          A          95
    4          C          80


No.Mhs : 123060201
Nama    : Fathimah

 Kode MKA    Kelas     NilaiUTS
    3          A          90
```

RECCURENCE RELATION (RELASI BERULANG)

Algoritma dan Pemrograman II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

RECCURENCE RELATION (RELASI BERULANG)

- Bab ini membahas pemakaian skema iterasi untuk PERSOALAN DERET yang rumusnya dapat dinyatakan dalam hubungan/relasi berulang, yaitu menyangkut masalah ketelitian penyajian bilangan pada komputer.
- Contoh :

  Barisan : 1,2,3,4,....,n
  3,5,7,9,....,n

  Deret : 1 + 2 + 3 + 4 + ... + n
  3 + 5 + 7 + 9 + ... + n

Barisan dan Deret

- Barisan : 1,2,3,4,5,...,n

```
Algoritma Contoh_Barisan
Deklarasi
  i,n : integer
Deskripsi
  input(n)
  i←1
  while (i<=n) do
    output(i)
    i←i+1
  endwhile
```

- Deret S = 1+2+3+4+5+...+n

```
Algoritma HitungDeret1
Deklarasi
  i,S,n : integer
Deskripsi
  input(n)
  i←1
  S←0
  while (i<=n) do
    output(i)
    S←S+i
    i←i+1
  endwhile
  output(S)
```

Contoh-contoh Algoritma Deret

HitungDeret2

S = 2+4+6+8+...+20+...+n

i	1	2	3	4	...	10	n
f(i)	2	4	6	8	...	20	2n

➔ f(i) = 2*i

x = 2*i

S = S + (2*i)

S = S + x

```
Algoritma HitungDeret2
Deklarasi
  i,S,x,n : integer
Deskripsi
  x←0
  S←0
  input(n)
  i←1
  while (i<=n) do
      x←2*i
      S←S+x
      output(x)
      i←i+1
  endwhile
  output(S)
```

BENTUK 1

Contoh HitungDeret2 dalam bahasa C :

```
#include<iostream.h>
main()
{
   int  i=1,S=0,x=0;

   cout << "S = ";
   while (i<=10)
   {
    x = 2*i;
    S +=x;
    cout << x << "+";
    i++;
   }
   cout << "\nJumlah deret S = " << S;
}
```

Output :

```
S = 2+4+6+8+10+12+14+16+18+20+
Jumlah deret S = 110
```

HitungDeret3

S = 3+5+7+9+11+13+15

i	1	2	3	4	...	n
f(i)	3	5	7	9	...	2n+1

➔ f(i) = 2*i+1

x = 2*i+1

S = S + (2*i+1)

S = S + x

```
Algoritma HitungDeret3
Deklarasi
  i,S,x : integer
Deskripsi
  i←1
  x←0
  S←0
  while (i<=7) do
      x←2*i+1
      S←S+x
      output(x)
      i←i+1
  endwhile
  output(S)
```

HitungDeret4

S = 2+5+10+17+26+...+101

i	1	2	3	4	5	...	n
f(i)	2	5	10	17	26	...	(n*n)+1

➔ f(i) = i*i+1

x = i*i+1

S = S + (i*i+1)

S = S + x

```
Algoritma HitungDeret4
Deklarasi
  i,S,x,n : integer
Deskripsi
  x←0
  S←0
  input(n)
  i←1
  while (i<=n) do
      x←i*i+1
      S←S+x
      output(x)
      i←i+1
  endwhile
  output(S)
```

HitungDeret5

S = -3+6-9+12-15+18

i	1	2	3	4	5	6	n
f(i)	-3	6	-9	12	-15	18	$3*n*(-1)^n$

➔ $f(i) = 3*i*(-1)^i$

$x = 3*i*(-1)^i$

$S = S + (3*i*(-1)^i)$

$S = S + x$

```
Algoritma HitungDeret5
Deklarasi
  i,S,x : integer
Deskripsi
  i←1
  x←0
  S←0
  while (i<=6) do
       x←3*i*(-1)^i
       S←S+x
       output(x)
       i←i+1
  endwhile
  output(S)
```

Cara lain : HitungDeret6

S = -3+6-9+12-15+18

i	1	2	3	4	5	6	n
f(i)	-3	6	-9	12	-15	18	3*n*tanda

➔ f(i) = 3*i*tanda

x = 3*i*tanda

S = S + x

tanda = -1 → untuk i ganjil

i mod 2 = 1

tanda = 1 → untuk i genap

i mod 2 = 0

```
Algoritma HitungDeret6
Deklarasi
  S,x,i,n,tanda:
                integer
Deskripsi
   i←1 x←0
   S←0 input(n)
   while (i<=n) do
      if(i mod 2=0)then
         tanda ← 1
      else
         tanda ← -1
      endif
      x← 3*i *tanda
      S←S+x
      output(x)
      i←i+1
   endwhile
   output(S)
```


BENTUK 2

HitungDeret7

$$S = \frac{1}{100}+\frac{1}{99}+\frac{1}{98}+\frac{1}{97}+\frac{1}{96}+\cdots+\frac{1}{2}+1$$

```
Algoritma HitungDeret7
Deklarasi
   i : integer
   S : real
Deskripsi
   S←0.0
   i←100
   while (i>=1) do
        output('1/',i,'+')
        S←S+1/i
        i←i-1
   endwhile
   output(S)
```

```
#include <iostream.h>
main()
{
 int i;
 float S;

 S=0.0;
 i=100;
 cout << "S = ";
 while (i>=1)
 {
        cout<<"1/"<<i<<"+";
        S=S+1/i;
        i=i-1;
 }
 cout <<"\nJumlah S = " << S;
}
```

HitungDeret8

$$S = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \cdots$$

```
Algoritma HitungDeret8
Deklarasi
   i,p,q,n : integer
   S : real
Deskripsi
   i←1
   p←1
   q←2
   S←0.0  input(n)
   while (i<=n) do
        output(p,'/',q)
        S←S+p/q
        p←p
        q←q*2
        i←i+1
   endwhile
   output(S)
```

Cara lain : HitungDeret8

$$S = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \cdots$$

```
Algoritma HitungDeret8
Deklarasi
   i,p,q,n : integer
   S,x : real
Deskripsi
   S←0.0
   input(n)
   i←1
   while (i<=n) do
        p← 1
        q←2^i
        x←p/q
        output(p,'/',q)
        S←S+x
        i←i+1
   endwhile
   output(S)
```


BENTUK 3

HitungDeret9

$$S = \frac{1}{2} - \frac{1}{4} + \frac{1}{8} - \frac{1}{16} + \frac{1}{32} - \cdots + \cdots$$

```
Algoritma HitungDeret9
Deklarasi
   i,p,q,n : integer
   S : real
Deskripsi
   i←1
   p←1
   q←2
   S←0.0   input(n)
   while (i<=n) do
        output(p,'/',q)
        S←S+p/q
        p←-1*p
        q←q*2
        i←i+1
   endwhile
   output(S)
```

Cara lain: HitungDeret10

$$S = \frac{1}{2} - \frac{1}{4} + \frac{1}{8} - \frac{1}{16} + \frac{1}{32} - \cdots + \cdots$$

```
Algoritma HitungDeret10
Deklarasi
   i,p,q,n : integer
   S,x : real
Deskripsi
   i←1
   input(n)
   S←0.0
   while (i<=n) do
        p← 1
        q← 2^i
        if (i mod 2 = 0)
            tanda ← -1
        else
            tanda ← 1
        x←p/q * tanda
        S←S+x
        i←i+1
   endwhile
   output(S)
```

BENTUK 4

Latihan

1. S = -3-6-9-12+15-18-...-...-...+...
2. S = 1-3+5-7+9-11+...+(2*i-1)*(-1)$^{i-1}$
3. S = 1-1/2+1/3-1/4+...+1/99-1/100
4. S = -1/2-1/4-1/6+1/8-...-...-...+...
5. S = 2/3+4/9-8/27+16/81+...-...

RECURSIVE

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

RECURSIVE

- Recursive is a tool / way to solve the problem in a function or procedure that calls itself.
- Definitions under Niclaus Wirth :

  "An object is said be recursive if it partially or is Consist defines in terms of itself"
- mathematical calculations (eg function factorial and Fibonacci numbers)

Factorial

- Function factorial of a positive integer n is defined as follows :

  n!= n.(n-1)! , jika n>1 → rekursive

  n!= 1 , jika n=0, 1 → basis

- example :

  3!= 3. 2!

  3!= 3. 2. 1!

  3!= 3. 2. 1

  3!= 6

We can write factorial timer function like this

```
1.   int Faktorial(int n)
2.  {
3.      if ((n == 0) || (n == 1 ))
4.           return (1);
5.      else
6.           return (n * Faktorial(n-1));
7.  }
```

- On line 3 of the above functions , n checked value equal to 0 or 1,
  - if yes , then the function returns a value of 1 {row 4} ,
  - if not , the function returns a value of n * factorial (n-1)
- Line { 6} this is where the recursive process, note this factorial function calls itself , but with parameters (n-1)

Fibonacci Numbers

- Another function that can be converted into a recursive form is the calculation of Fibonacci . Fibonacci numbers can be defined as follows :

  $f_n = f_{n-1} + f_{n-2}$ untuk $n > 2$

  $f_1 = 1$

  $f_2 = 1$

  Here is a Fibonacci sequence of numbers ranging from n=1

  1 1 2 3 5 8 13 21 34

Fibonacci Algorithm used

```
Function Fibonacci(input n:integer) → integer
Deklarasi Lokal
   {tidak ada}
Deskripsi
   If (n =1 or n=2) Then
      return (1)
   Else
      return (Fibonacci(n-1)+Fibonacci(n-2))
   Endif
```

Example

- For the size of n = 4 , Fibonacci calculation process can be carried out as follows :

$$f_4 = f_3+f_2$$
$$f_4 = (f_2+f_1) + f_2$$
$$f_4 = (1+1) +1$$
$$f_4 = 3$$

Combination

```
Function Kombinasi (input n, r : integer) → real
Deklarasi
 If (n < r) Then
  return (0)
 Else
  return (Faktorial(n)/(Faktorial(r)*Faktorial(n-r)))
 Endif
```

Permutation

```
Function Permutasi (input n, r : integer) → real
Deklarasi
    {tidak ada}
 Deskripsi
    If (n< r) Then
          return (0)
    Else
          return (Faktorial(n) / Faktorial(n-r))
    Endif
```

Calculating series using recursion

- Calculating series S = 1+2+3+4+5+...+n

S = 1+2+3+4+5

= 5+4+3+2+1

S(1) = 1

S(2) = 2+1 = 2 + S(1)

S(3) = 3+2+1 = 3 + S(2)

S(4) = 4+3+2+1 = 4 + S(3)

S(n) = 1+2+3+4+5+...+(n-1)+n

= n+(n-1) + ... + 4+3+2+1

= n+S(n-1) → Rekursi

S=1+2+3+4+5+…+n

```
Function S(input n:integer) → integer
Deklarasi Lokal
   {tidak ada}
Deskripsi
  If (n==1) Then
     return (1)
  Else
     return (n + S(n-1))
  Endif
```

in C++ (full)

```
#include<iostream.h>
int S(int n);
main()
{
  int n;
  cout << "Masukkan n = "; cin >> n;
  cout << "Deret S=1+2+3+4+5+...+n \n";
  cout << "Jumlah deret S = " << S(n);
}
int S(int n)
{
      if (n == 1)
        return (1);
      else
        return (n + S(n-1));
}
```

Exercise

1. Create a program to calculate a series of S = 2 + 4 + 6 + 8 + 10 + ... + 2n using recursion function
2. Create a program to calculate a series of S = 2 + 4 + 8 + 16 + 32 + ... + 2^n using recursion function

No.1 S=2+4+6+8+10+…+2n

```
Function S(input n:integer) → integer
Deklarasi Lokal
   {tidak ada}
Deskripsi
  If (n==1) Then
     return (2)
  Else
     return (2*n + S(n-1))
  Endif
```

No.2 S=2+4+8+16+32+…+2^n

```
Function S(input n:integer) → integer
Deklarasi Lokal
   {tidak ada}
Deskripsi
  If (n==1) Then
     return (2)
  Else
     return (2^n + S(n-1))
  Endif
```

SEARCHING

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

SEARCHING

- Search is required to seek specific information from a table at the time the exact location of the information previously unknown . Data in the tables are usually stored by using the data type Array
- Array makes it possible to store the value of the same type . Operations that are common in the array is Sequential Search and Binary search.
- Differences between this technique lies in state data.

Sequential Search

- Sequential search can be used in data in a state of random or sequential and that has been ordered

- Sequential search is often called Linear search using the following principles

  data that is compared one by one in sequence with data sought .

Sequential Search

1. Sequential Search on the Array data elements Not Sorted
   a. Methods without Sentinel
   b. Methods with sentinel

2. Sequential Search on the Array data elements Already Ascending
   a. Sequential Method Without Sentinel
   b. Metode Sequential Search Dengan Sentinel

Sequential search process data has not been ordered without sentinel :

- This search is basically just doing the repetition of elements of the 1st up to the amount of data .
- on each repetition , as compared to the data - i to be searched,
- if the same means data has been found,
- otherwise if no repetition until the end of the same , meaning no data.

Sample Program SeqSearch_BelumUrut_nonSentinel

```
/* SeqSearch_BelumUrut_nonSentinel
   Array X is assumed to already exist and contain data
   that has not been    sequence, the value sought is y
   and there is only one */

#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
   int X[10]={20,50,10,30,90,60,70,80,40,100};
   boolean found;
   int i,y;
   cout << " value sought = ";
   cin >> y;
```

```
  found=false;
  i=0;
  while ((i<10) && (!found))
  {
    if (X[i]==y)
     found = true;
    else
     i = i + 1;
  }
  if(found)
  cout << y <<" found in the index array " <<i;
  else
  cout << y <<" not in the Array ";
}
```

- Another way to Sequential Search on the data array X[n] , is to provide a single point after the last one, ie at X[n+1] and save the price sought (eg y) on the position. Sought by the position of the last element is called sentinel .

Sequential search process data has not been ordered by the sentinel :

- This search is basically the same as the sequential search process data has not been ordered without sentinel ie repetition of elements of the 1st up to the amount of data.
- on each repetition , as compared to the data - i to be searched, if the same means data has been found
- the difference with that without sentinel is when the data is found but these data are sentinel means no data.

Sample Program SeqSearch_BelumUrut_Sentinel {cara1}

```
/* SeqSearch_BelumUrut_Sentinel{cara1}
  assumed Array X [ 0..10 ] is already there , and
    the index to 0..9 already contain data that is not
    in order, the value sought is y and there is only
    one , y is placed in the index to 10 */
#include <iostream.h>
main()
{
 int X[11]={20,50,10,30,90,60,70,80,40,100};
 int i,y;
 cout << " value sought = ";
 cin >> y;
 X[10]=y;
```

```
i=0;
 while (X[i]!=y)
    i=i+1;
 if (i>9)
    cout << "no " << y << " in the Array";
 else
    cout << y << " found in the Array at index " <<
   i;
}
```

Sample Program SeqSearch_BelumUrut_Sentinel {cara2}

```
/* SeqSearch_BelumUrut_Sentinel {cara2}
   Array X is assumed to already exist and contain
   data that has not been    sequence, the value
   sought is y and there is only one */
#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
 int X[11]={20,50,10,30,90,60,70,80,40,100};
 int i,y;
 boolean found;
 cout << " value sought = ";
 cin >> y;
 X[10]=y;
```

```
found=false;
 i=0;
 while (!found)
 {  if (X[i]==y)  found=true;
    else  i=i+1;
 }
 if (i==10)
   cout << "no " << y << " in the Array";
 else
   cout <<  y << " array is found in the index " <<
  i;
}
```

Sequential search process of data already ordered :

- Starting from the first element in the array were compared with the element in question . If the elements in the array is smaller than the element in question , the search continued. If it is larger, the search is stopped, and it is certain that the element in question did not exist .
- If used with a sentinel search ways (the element in question is inserted in the index after the last data), and the element in question is greater than the last data in the array so that the data sought is the same as the sentinel of data it can be concluded that the data can not be found.

Sample Program SeqSearch_Urut_NonSentinel

```
/* SeqSearch_SudahUrut_NonSentinel
   Array X is assumed to already exist and contain
   data that has been ordered , value sought is y and
   there is only one */
#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
 int X[10]={10,20,30,40,50,60,70,80,90,100};
 int i,y;
 boolean found;
 cout << " value sought = ";
 cin >> y;
 found=false;
```

```
i=0;
 while ((i<10) && (!found) && (y>=X[i]))
 {if (X[i]==y)
      found=true;
  else
      i=i+1;
 }

if (found)
   cout <<  y << " found in the array at index" <<
  i;
 else
   cout << "no " << y << " in the Array";
}
```

Sample Program SeqSearch_Urut_Sentinel {cara1}

```
/* SeqSearch_SudahUrut_Sentinel {cara 1}
assumed Array X [ 1..nmax ] already exists and the
index 1..n already contain data that has been ordered
, the value sought is y and there is only one*/

#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
 int X[11]={10,20,30,40,50,60,70,80,90,100};
 int i,y;
 boolean found;
 cout << « value sougth = "; cin >> y;
```

```
 X[10]=y;
 found=false;
 i=0;
 while ((!found) & (X[i]<y))
   i=i+1;  //no checking met or not
 if (i>9)
    cout << " no " << y << " in the Array";
 else
    if (X[i]==y)
      cout << y <<" Array is found in the index " << i;
    else
      cout << "no " << y << " in the Array";
}
```

Sample Program SeqSearch_Urut_Sentinel {cara2}

```
/* SeqSearch_SudahUrut_Sentinel {cara 2}
assumed Array X [ 0..10 ] 1..9 index of existing and
already contain data that has been ordered , the
value sought is y and there is only one */
#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
 int X[11]={10,20,30,40,50,60,70,80,90,100};
 int i,y;
 boolean found;
 cout << « value sought = "; cin >> y;
```

```
X[10]=y;
found=false;
i=0;
while ((!found) & (X[i]<=y))
{if (X[i]==y)
    found=true;
  else
    i=i+1;
}
if (i==10)
  cout << "no " << y << " in the Array";
else
  if (found)
  cout <<  y << " found in the Array at index " << i;
  else
     cout << "no " << y << " in the Array";
}
```

Binary Search

- One of the conditions binary search (Binary Search) do is data already in a state of sequences .

Steps of binary search is as follows :

1. initially taken the starting position =1 and end position = n
2. then we find the position of the data center by the formula = center position (starting position + end position) div 2 then the data were compared with data sought middle
    1. if the same data is found, the process is complete .
    2. if smaller , the process is carried back but the end position is considered the same as the center position - 1.
    3. if greater , the process was repeated , but the starting position is considered the same as the center position + 1.
3. Repeat step 2 until the data is found , or not found.
4. This binary search will end if the data is found or the starting position is greater than the final position . If the starting position is already larger than the end position means the data is not found .

Example

- Suppose we want to find the number 14 on a set of data follows:

1	2	3	4	5	6	7	8	9
3	7	10	12	13	14	20	24	29

awal tengah akhir

Answer:

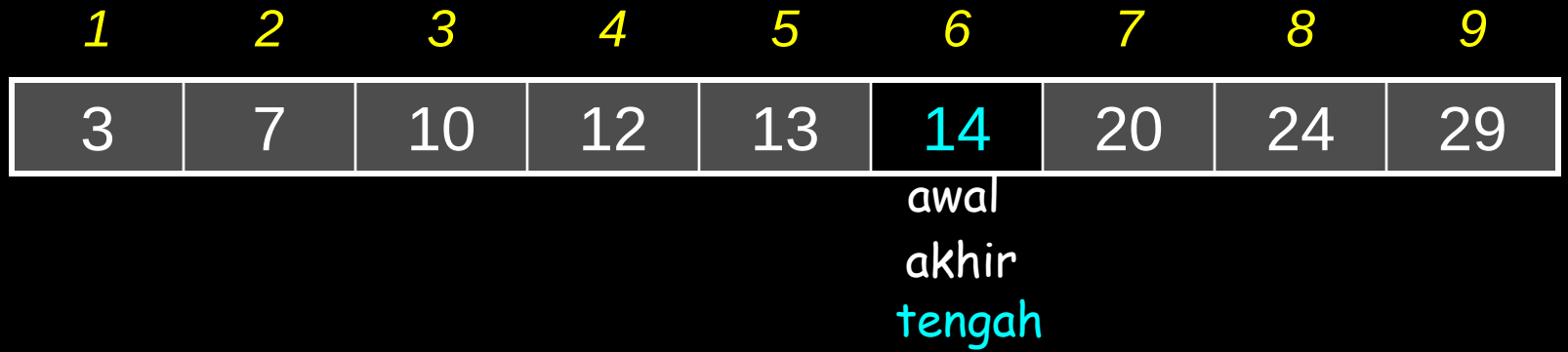
1. originally requested the data center, with the formula tengah = (awal+akhir) div 2 = (1+9) div 2 = 5, means the middle of the data is the data 5th , with a value = 13
2. the data we are looking for is 14 , compare the value of 14 with the data central

3. because 14 > 13 , means the process followed by starting position is considered the same as the center position+1 or 6 positions :

1	2	3	4	5	6	7	8	9
3	7	10	12	13	14	20	24	29
					awal	tengah		akhir

4. Data sought a new center , obtained from formula middle = (initial + final) div 2 = (6 + 9) div 2 = 7 , meaning the data is the data the new middle 7th, which is 20.
5. the data we are looking for is 14 , 14 compare with the center data

6. because 14 < 20 , means the process continues , with the final position is considered the same as the center position-1 or 6 position.



7. Searchable data back amid new formula derived from the middle = (initial + final) div 2 = (6 + 6) div 2 = 6 , meaning that data is the new middle is data -6, which is 14 the data we are looking for is 14 , compared with
8. the data center, it turns out the same, so the data found on the index sixth

In general, the binary search algorithm , is as follows :

* Data sorted in advance , the data stored in the data array, x is the value sought

1. awal ← 1
2. akhir ← N
3. ketemu ← false
4. while (awal<=akhir) and (not ketemu) do baris 5 to 8.
5. tengah ← (awal+akhir) div 2
6. if (data [tengah] = x) then ketemu ← true
7. if (x < data [tengah]) then akhir ← tengah-l
8. if (x > data [tengah] then awal ← tengah+1.
9. if (ketemu) then tengah is an index of the data sought, otherwise the data can not be found.

Example usage binary search

```
/* BinSearch_SudahUrut
Array X is assumed to already exist and contain data that
has been ordered , value sought is y and there is only
one*/
#include <iostream.h>
typedef enum boolean {false=0,true=1};
main()
{
 int X[10]={10,20,30,40,50,60,70,80,90,100};
 int i,y,j,k;
 boolean found;
 cout << "value sought = ";
 cin >> y;
```

```
found = false;
i=0;
j=10;
while ((!found) & (i <= j))
{
       k=(i+j)/2;
       if (y == X[k])
             found=true;
       else
             if (y<X[k])
                   j=k-1;    //i tetap
             else
                   i=k+1; //j tetap
}
if (found)
      cout<< y<<"found in the Array at index" << k;
else
      cout << "no " << y << " in the Array";
}
```

Exercise :

1. Illustrate search values below 25 on a data set .
   Use binary search method

12	15	4	25	45	10	19	9	32	33

2. Illustrate search values below 20 on a data set .
   Use binary search method .

12	15	4	25	45	10	19	9	32	33

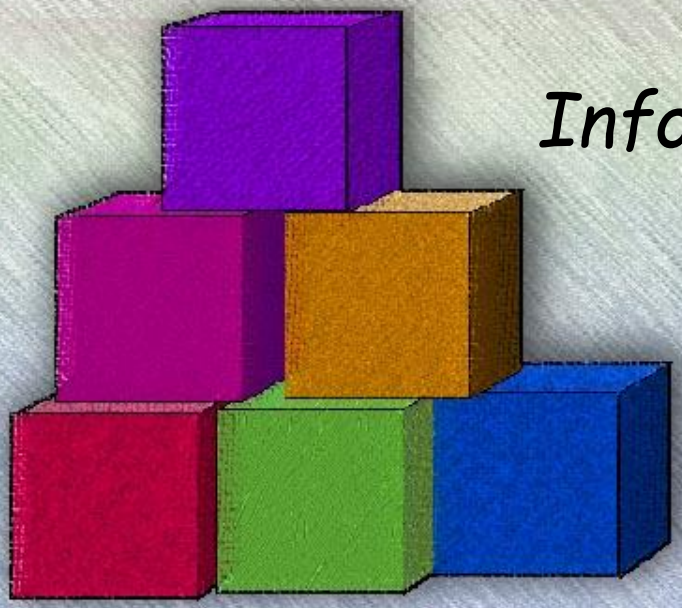
3. Write a program to find the type of data recording using one of the methods of search . 10. The number of data recording mode has four fields , namely :
   - Nomorinduk , of type integer
   - Name , type string
   - Address , type string
   - Class , type char (can be worth 'A'...'Z')

   Data sought through nomorinduknya . When the data sought found, show three other fields .

SORTING

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk



SORTING

- Sorting is a process of sorting the data previously arranged randomly or irregularly become orderly and organized according to a certain rule .
- Usually the sorting is divided into two, namely :
  - ascending (sorting of characters / numbers to the character of the small / large numbers).
  - descending (sorting of characters / large numbers to characters / small numbers).

Sorting Method

Sorting methods directly:

- *Exchange selection (Bubble Sort)*
- *Straight Selection Sort*
- *Straight Insertion Sort*

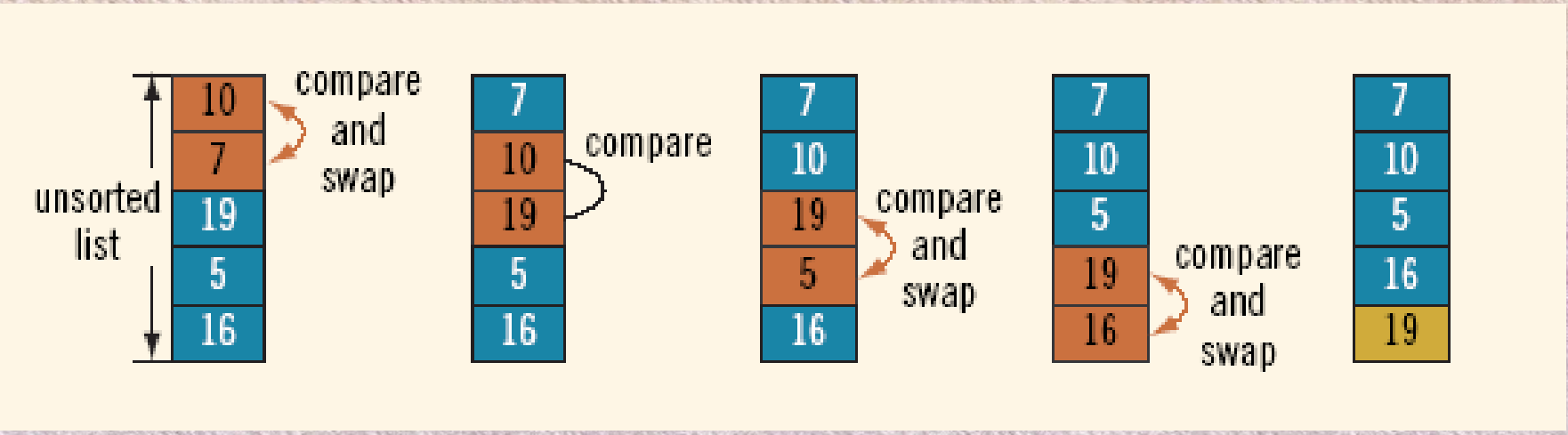
Sorting methods indirectly:

- Shell Sort
- Quick Sort
- Merge Sort

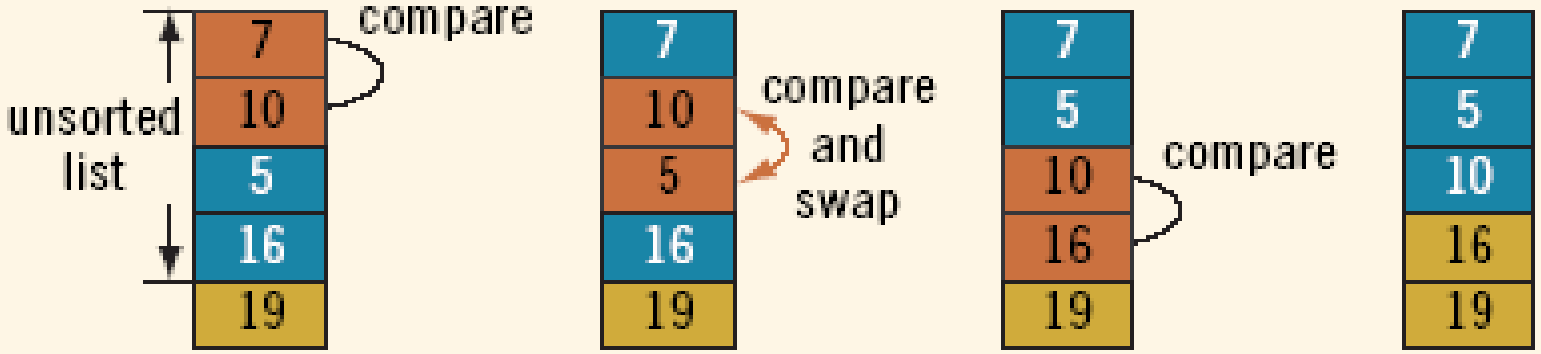
Bubble Sort Algorithm

- Bubble sort algorithm:
  - Suppose `list[0...N – 1]` is a list of n elements, indexed `0` to `N – 1`
  - We want to rearrange; that is, sort, the elements of list in increasing order
  - The bubble sort algorithm works as follows:
    - In a series of `N – 1` iterations, the successive elements, `list[index]` and `list[index + 1]` of list are compared
    - If `list[index]` is greater than `list[index + 1]`, then the elements `list[index]` and `list[index + 1]` are swapped, that is, interchanged

Bubble Sort Algorithm (Cont'd)

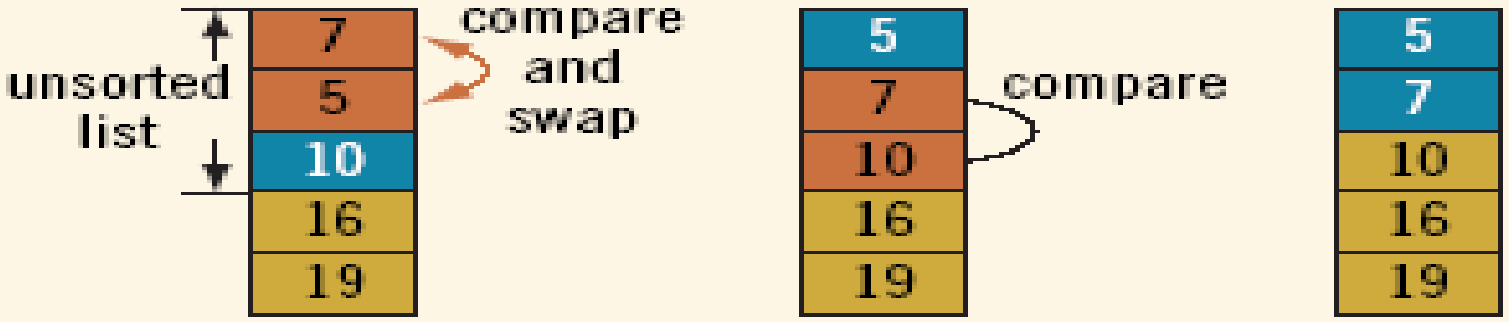


Elements of array *list* during the first iteration

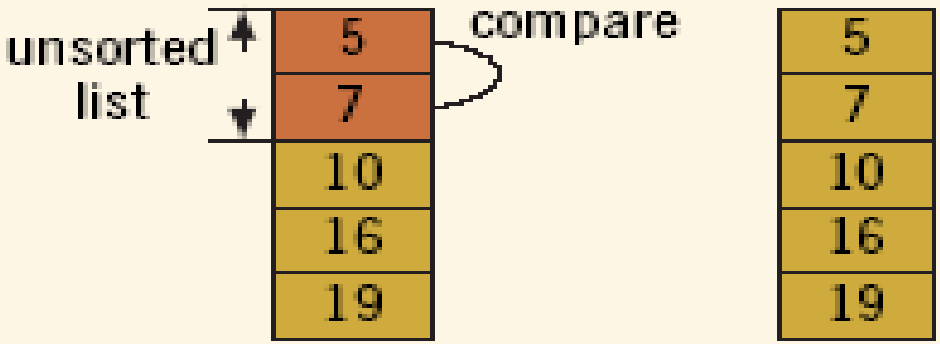


Elements of array *list* during the second iteration

Bubble Sort Algorithm (Cont'd)



Elements of array *list* during the third iteration



Elements of array *list* during the fourth iteration

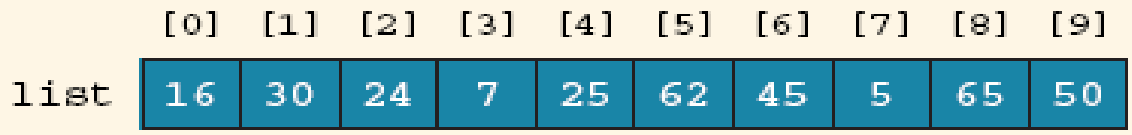
Bubble Sort Algorithm (Cont'd)

```
void bubbleSort(int[] list, int listLength) {
    int temp, counter, index;
    int temp;
    for (counter = 0; counter < listLength; counter++) {
        for (index = 0; index < listLength – 1 - counter; index++) {
            if(list[index] > list[index+1]) {
                temp = list[index];
                list[index] = list[index+1];
                list[index] = temp;
            }
        }
    }
} //end bubbleSort
```

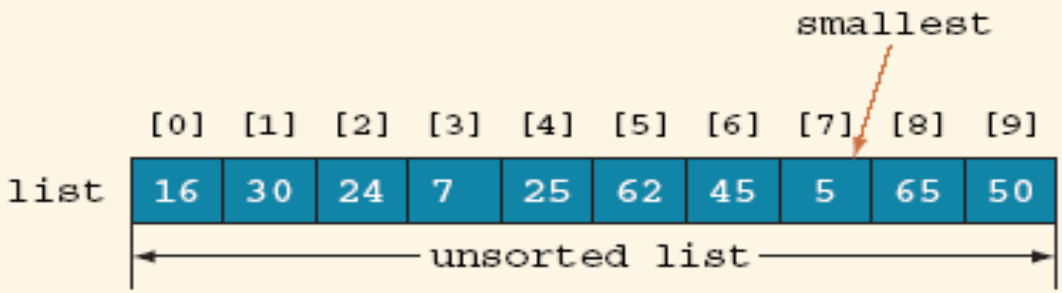
Selection Sort Algorithm

- List is sorted by selecting list element and moving it to its proper position
- Algorithm finds position of smallest element and moves it to top of unsorted portion of list
- Repeats process above until entire list is sorted

Selection Sort Algorithm (Cont'd)

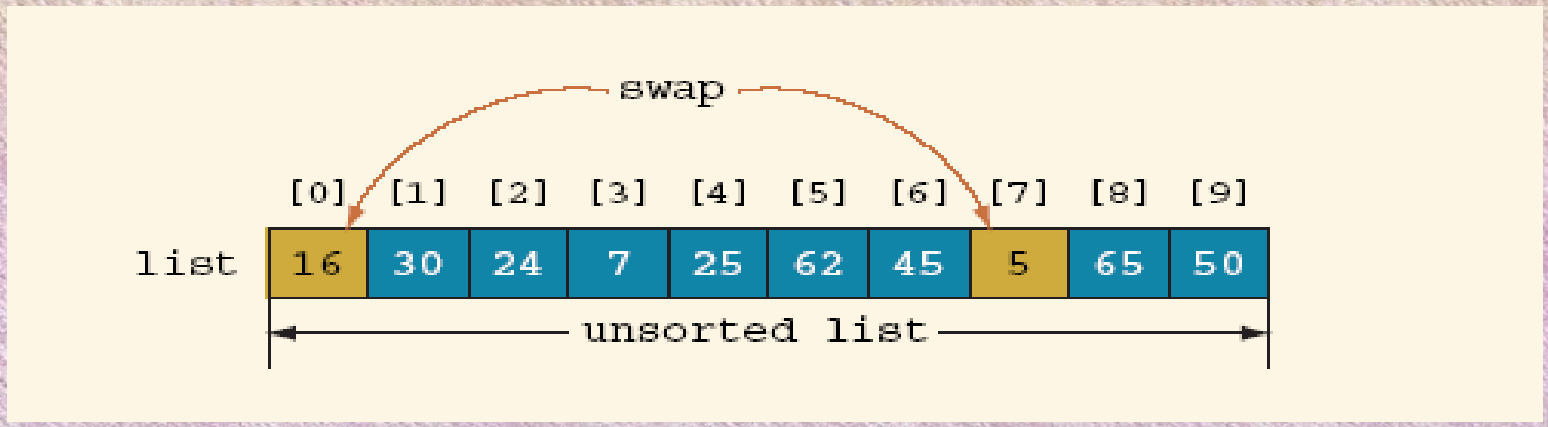


An array of 10 elements

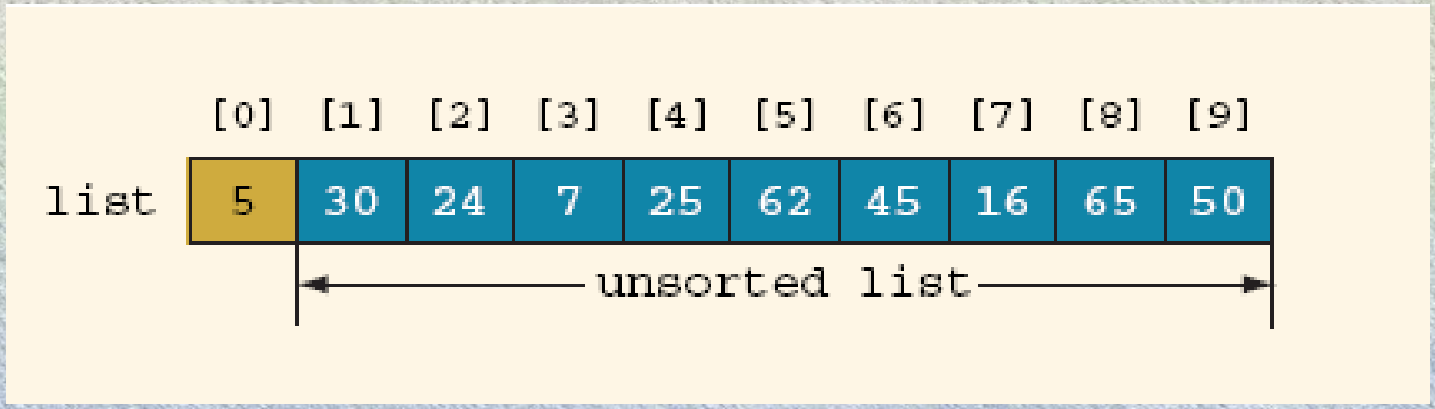


Smallest element of unsorted array

Straight Selection Sort Algorithm (Cont'd)



Swap elements *list[0]* and *list[7]*

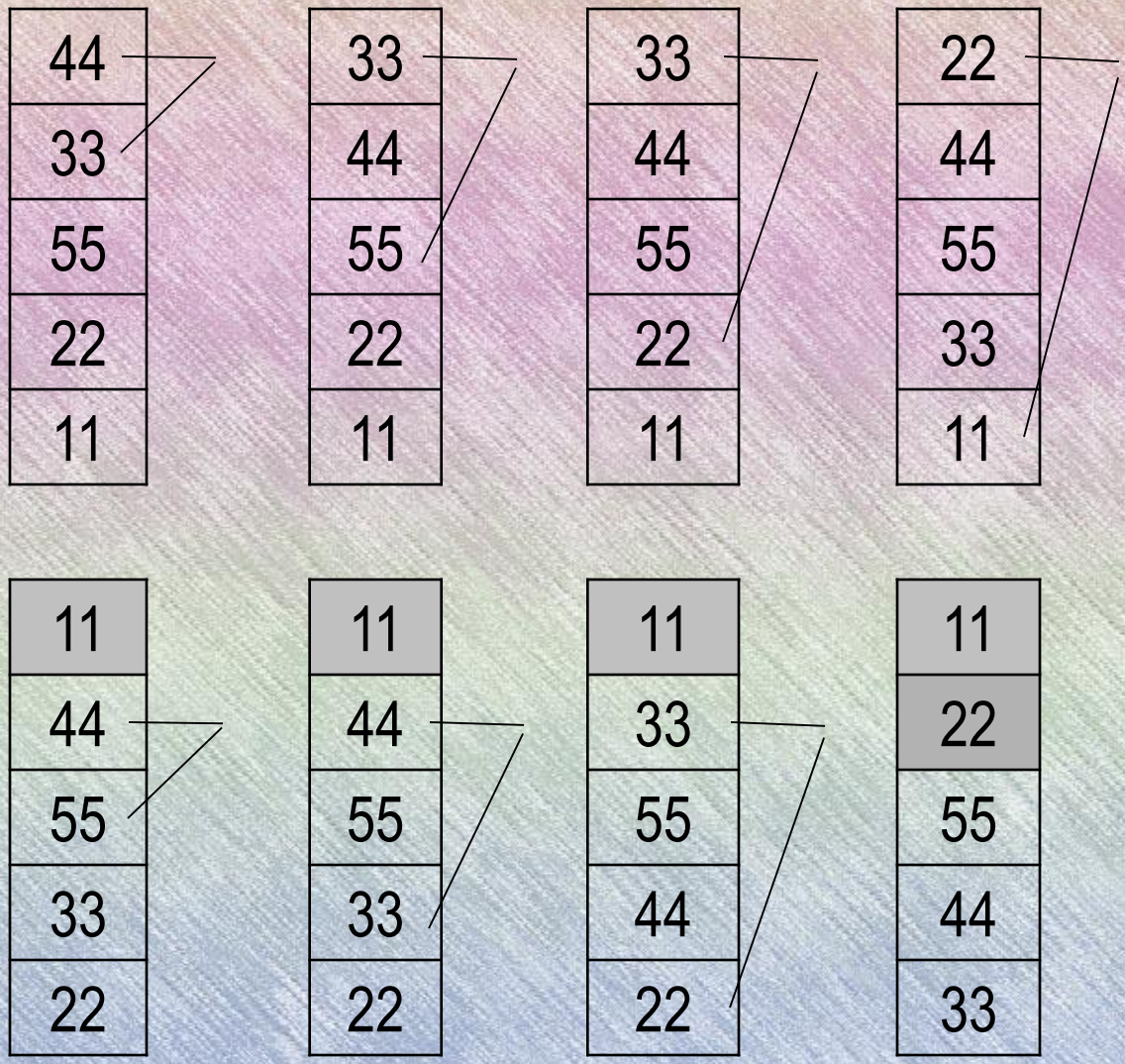


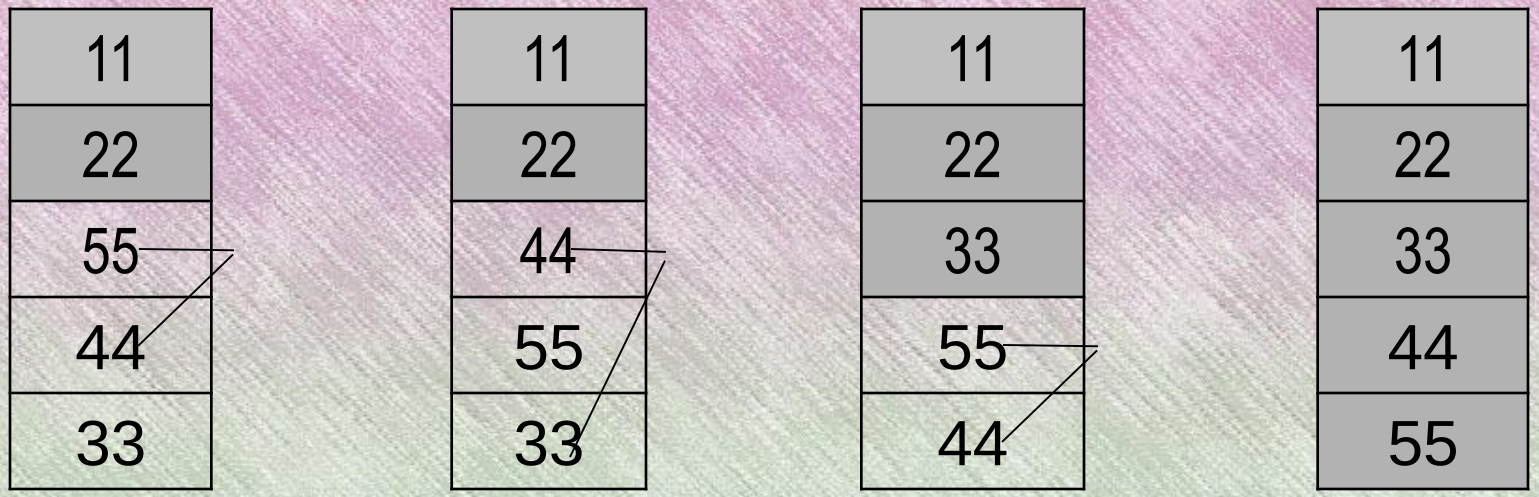
Array after swapping *list[0]* and *list[7]*

Straight Selection Sort steps:

1 : Read array elements are sorted (n)
2 : Do steps 3 through step 5 for i = 1 to n-1
3 : Specify the starting location smallest data
Mindeks = 1;
do step 4 for j=i+1 to n
4 : Search data is the smallest and note its location.
Test whether $A_{Mindeks} > A_j$?
If yes, note Mindeks = j
5 : Exchange value $A_{mindeks}$ with A_j
6 : finishing

Illustration Straight Selection Sort




11
22
55
44
33
11
22
44
55
33
11
22
33
55
44
11
22
33
44
55

Selection Sort Algorithm (Cont'd)

```
void selectionSort(int[] list, int listLength) {
    int index;
    int smallestIndex;
    int minIndex;
    int temp;

    for (index = 0; index < listLength – 1; index++) {
        smallestIndex = index;
        for (minIndex = index + 1;
             minIndex < listLength; minIndex++)
            if (list[minIndex] < list[smallestIndex])
                smallestIndex = minIndex;

        temp = list[smallestIndex];
        list[smallestIndex] = list[index];
        list[index] = temp;
    }
}
```

Straight Insertion Sort

Can be divided into 2 parts

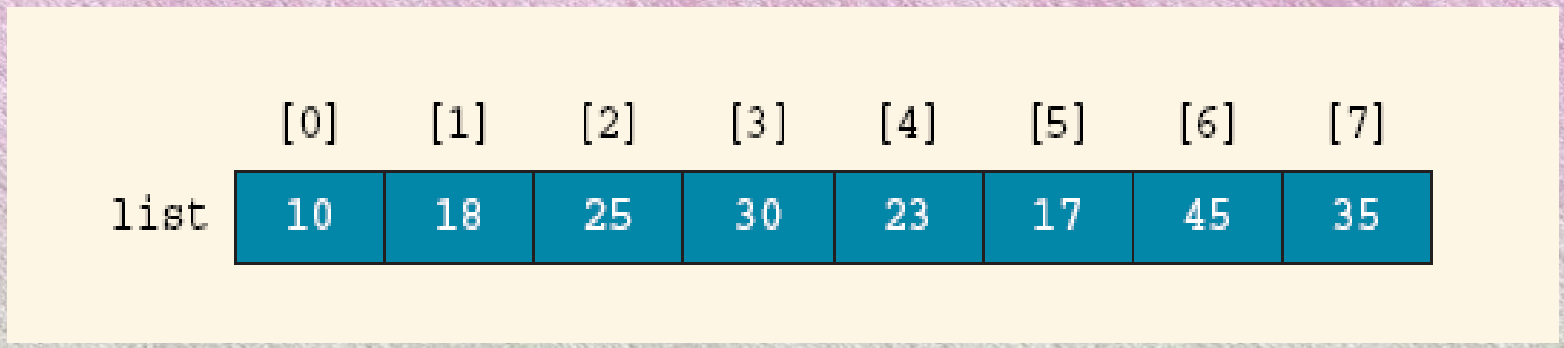
- The left portion of data has been sequenced (destination)
- Section to the right data has not been ordered (source)

Steps :

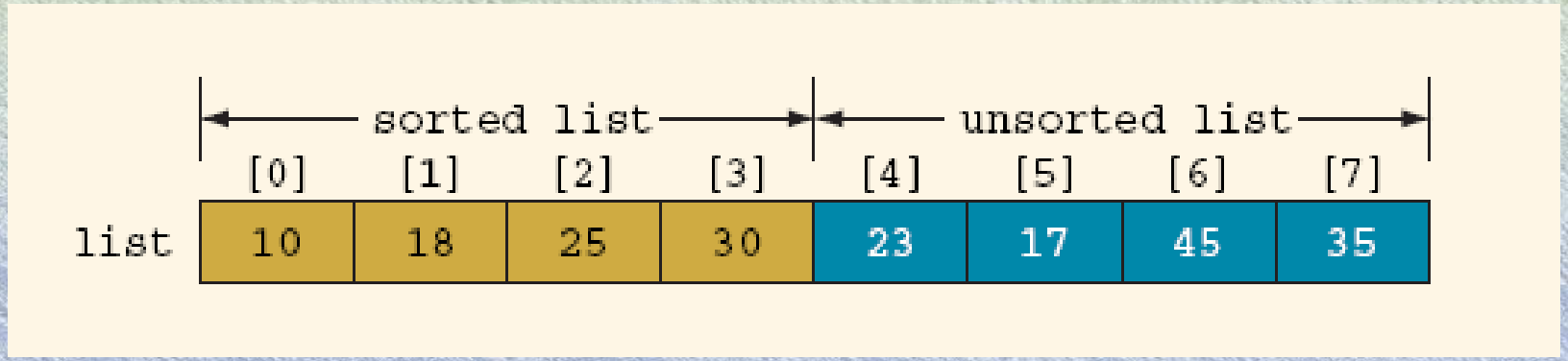
1 : Read array element to be sorted (n)
2 : Do step 3 to 6 for i : 1 to n-1
3 : Specify the element to be inserted (Temp = A [i] ; j = i-1;)
4 : Do step 5 for temp <A [j] and j >= 0;
5 : A [j+1]= A[j] ; j =j-1;
6 : Place the elemen ts A [j+1] = Temp;
7 : Finishing

Insertion Sort Algorithm

- The insertion sort algorithm sorts the list by moving each element to its proper place

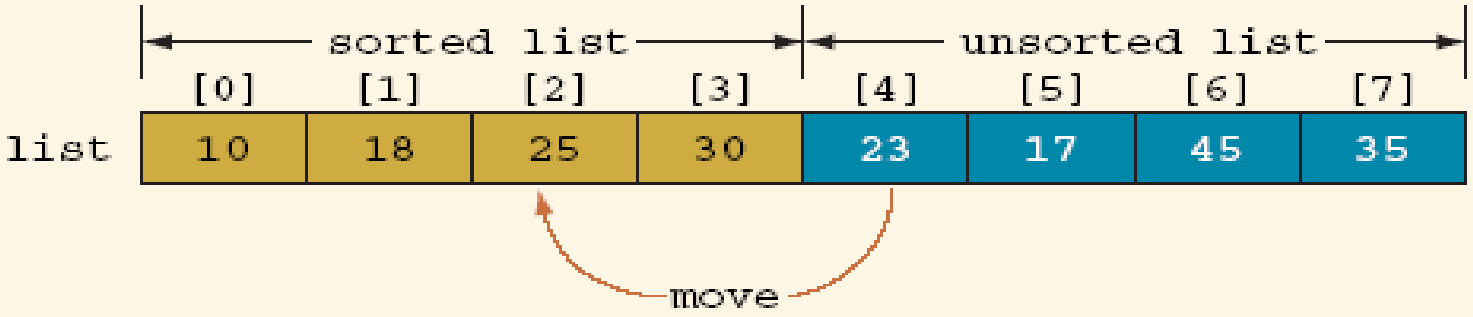


Array *list* to be sorted

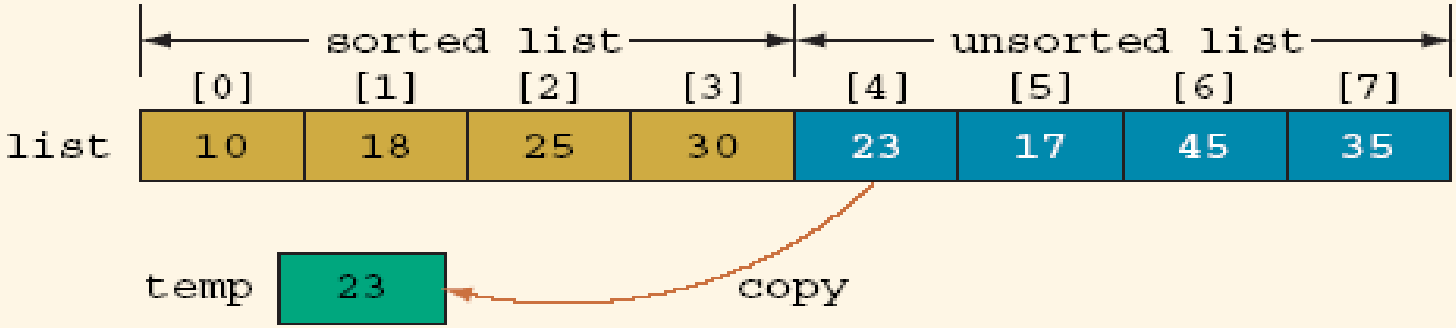


Sorted and unsorted portions of the array *list*

Insertion Sort Algorithm (Cont'd)

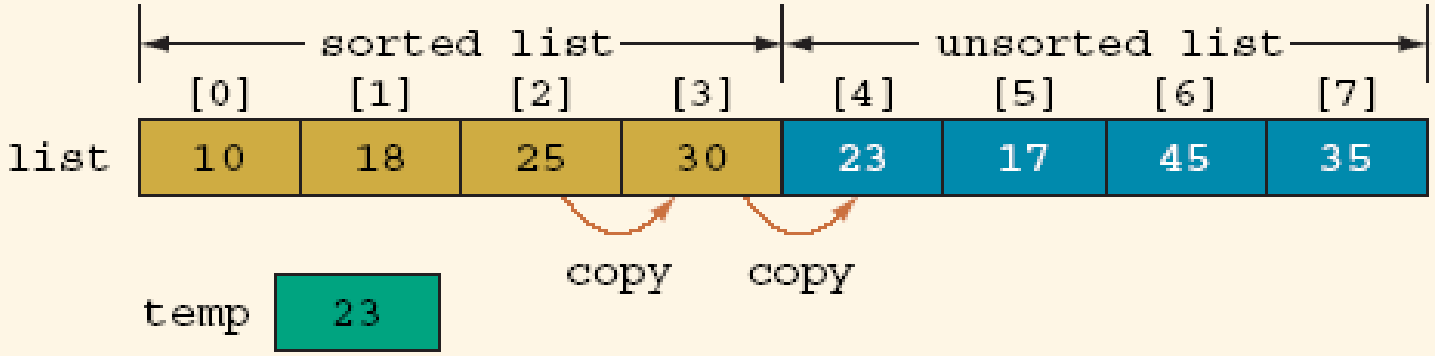


Move *list[4]* into *list[2]*

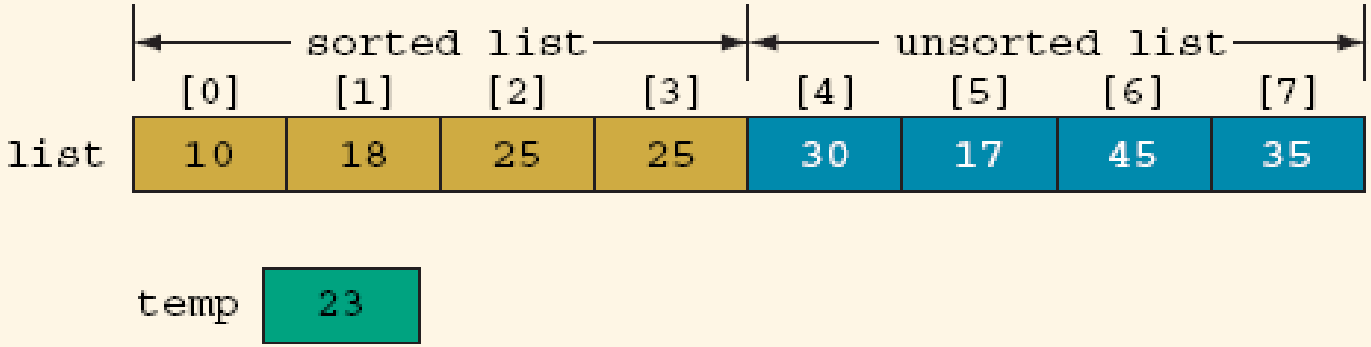


Copy *list[4]* into *temp*

Insertion Sort Algorithm (Cont'd)

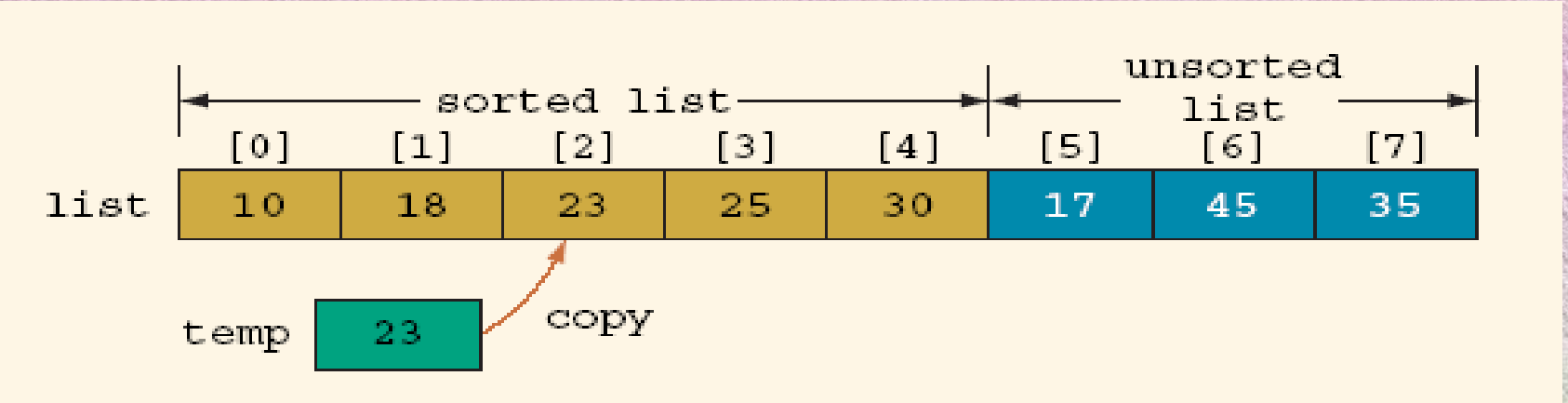


Array *list* before copying *list[3]* into *list[4]*, then *list[2]* into *list[3]*



Array *list* after copying *list[3]* into *list[4]*, and then *list[2]* into *list[3]*

Insertion Sort Algorithm (Cont'd)



Array *list* after copying *temp* into *list[2]*

Illustration of Insertion Sort

A [0]	A [1]	A [2]	A [3]	A [4]	A [5]
4	7	9	5	8	6
4	7	9	5	8	6
4	**7**	9	5	8	6
4	**7**	**9**	5	8	6
4	7	**9**	**5**	8	6
4	**7**	**5**	9	8	6
4	**5**	7	9	8	6
4	5	7	**9**	**8**	6
4	5	**7**	**8**	9	6
4	5	7	8	**9**	**6**
4	5	7	**8**	**6**	9
4	5	**7**	**6**	8	9
4	**5**	**6**	7	8	9

Straight Insertion Sort Algorithm

```
Deklarasi
   I,J,K,N : Integer
   Temp : real
   A : array [1..20] of real
Deskripsi
   Input(N) {maksimal N=20}
   K traversal [1..N]
     Input (Af) {masukkan data sebanyak N}
   I traversal [2..N]
     Temp ← A1
   J ← I-1
   While (temp <Aj) and (J>=1) do
     Aj+1 ← Aj
     J ←J-1
   Endwhile
   Aj+1 ← Temp
```

Insertion Sort Algorithm (Cont'd)

```
void insertionSort(int[] list, int listLength) {
    int firstOutOfOrder, location;
    int temp;
    for (firstOutOfOrder = 1;
         firstOutOfOrder < listLength;
         firstOutOfOrder++)
    if (list[firstOutOfOrder] < list[firstOutOfOrder - 1]) {
        temp = list[firstOutOfOrder];
                        location = firstOutOfOrder;
        do {
            list[location] = list[location - 1];
            location--;
        }
        while(location > 0 && list[location - 1] > temp);
        list[location] = temp;
    }
} //end insertionSort
```

Shell Sort

- Created by Donald Shell in 1959
- Wanted to stop moving data small distances (in the case of insertion sort and bubble sort) and stop making swaps that are not helpful (in the case of selection sort)
- Start with sub arrays created by looking at data that is far apart and then reduce the gap size

Shell Sort

This method utilizes a pair exchange elements to achieve a state sequence. Two elements exchanged with a certain distance.

Formula :
- Distance first = N / 2
- distance next= distance previous / 2

Example :

24	46	11	26	57	38	27	20	17
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

So that :

- Distance first = 9 / 2 = 4
- second distance = 4 / 2 = 2
- third distance = 2 / 2 = 1

Illustration of Shell Sort

Jarak	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
Awal	24	46	11	26	57	38	27	20	17
Jarak = 9div2=4	24	46	11	26	57	38	27	20	17
	24	38	11	26	57	46	27	20	17
	24	38	11	26	57	46	27	20	17
	24	38	11	20	57	46	27	26	17
	24	38	11	20	17	46	27	26	57
	17	38	11	20	24	46	27	26	57

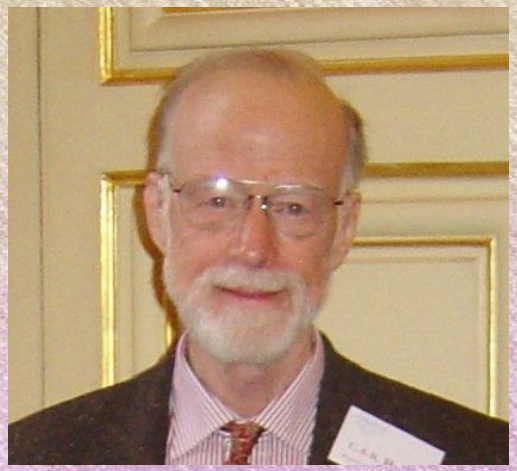
Jarak = 4div2=2	**17**	38	**11**	20	24	46	27	26	57
	11	**38**	**17**	**20**	24	46	27	26	57
	11	**20**	**17**	**38**	**24**	46	27	26	57
	11	20	**17**	**38**	**24**	**46**	27	26	57
	11	20	17	**38**	**24**	**46**	**27**	26	57
	11	20	17	38	**24**	**46**	**27**	**26**	57
	11	20	17	**38**	24	**26**	27	**46**	57
	11	**20**	17	**26**	24	38	27	46	57
	11	**20**	17	**26**	24	38	**27**	46	**57**
	11	20	17	26	24	38	**27**	46	**57**

Distance = div2=1	**11**	**20**	17	38	24	26	27	46	57
	11	**20**	**17**	38	24	26	27	46	57
	11	**17**	**20**	38	24	26	27	46	57
	11	**17**	**20**	**38**	24	26	27	46	57
	11	17	**20**	**38**	**24**	26	27	46	57
	11	17	**20**	**24**	38	26	27	46	57
	11	17	**20**	**24**	**26**	**38**	27	46	57
	11	17	20	24	**26**	**38**	**27**	46	57
	11	17	20	24	**26**	**27**	**38**	46	57
	11	17	20	24	**26**	**27**	**38**	**46**	57
	11	17	20	24	26	27	**38**	**46**	**57**
Result	11	17	20	24	26	27	38	**46**	**57**

Shell Sort Code

```
void shellsort(Comparable[] list)
{  Comparable temp; boolean swap;
   for(int gap = list.length / 2; gap > 0; gap /= 2)
      for(int i = gap; i < list.length; i++)
      {   Comparable tmp = list[i];
          int j = i;
          for( ; j >= gap &&
                 tmp.compareTo( list[j - gap] ) < 0;
                 j -= gap )
             list[ j ] = list[ j - gap ];
          list[ j ] = tmp;
       }
}
```

Quicksort



- Invented by C.A.R. (Tony) Hoare
- A divide and conquer approach that uses recursion

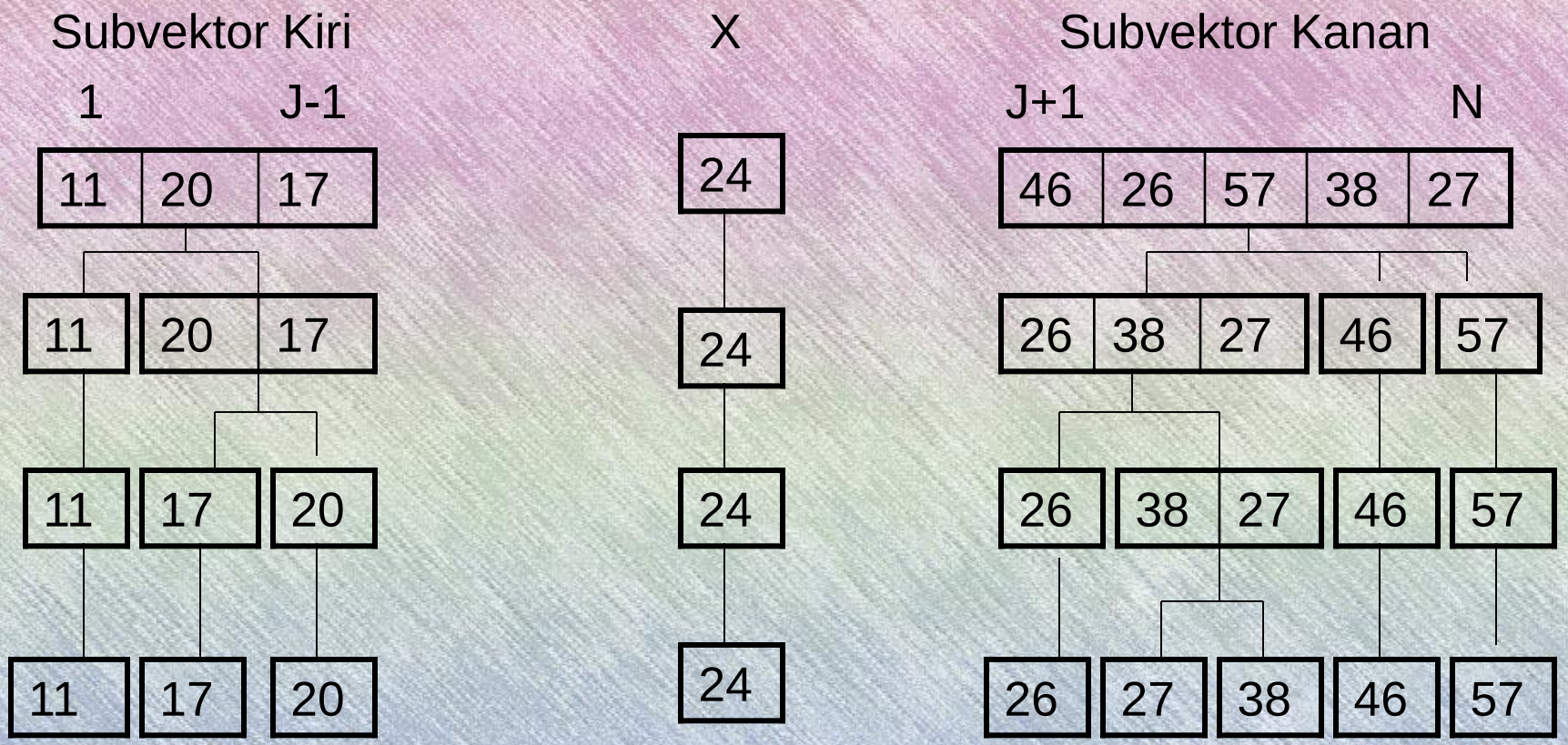
1. If the list has 0 or 1 elements it is sorted
2. otherwise, pick any element p in the list. This is called the pivot value
3. Partition the list minus the pivot into two sub lists according to values less than or greater than the pivot. (equal values go to either)
4. return the quicksort of the first list followed by the quicksort of the second list

Quick Sort

- Sering disebut dengan "partion exchange" sort.
- Langkah-langkah :
  - Misal vektor A yang memiliki N elemen.
  - Dipilih sembarang elemen, biasanya elemen pertama, misal sebut saja X.
  - Kemudian, semua elemen tersebut dengan menempatkan X pada posisi J sedemikian rupa sehingga :
    - elemen 1 s/d J-1 memiliki nilai lebih kecil dari X dan
    - elemen ke J+1 s/d N memiliki nilai lebih besar dari X.
  - Dengan demikian, terdapat dua buah subvektor.
- Contoh :

1								N
24	46	11	26	57	38	27	20	17

Ilustrasi Quick Sort



```
void swapReferences( Object[] a, int index1, int index2 )
{   Object tmp = a[index1];
    a[index1] = a[index2];
    a[index2] = tmp;
}

void quicksort( Comparable[] list, int start, int stop )
 {    if(start >= stop)
        return; //base case list of 0 or 1 elements

      int pivotIndex = (start + stop) / 2;

      // Place pivot at start position
      swapReferences(list, pivotIndex, start);
      Comparable pivot = list[start];

      // Begin partitioning
      int i, j = start;

      // from first to j are elements less than or equal to pivot
      // from j to i are elements greater than pivot
      // elements beyond i have not been checked yet
      for(i = start + 1; i <= stop; i++ )
      {   //is current element less than or equal to pivot
          if(list[i].compareTo(pivot) <= 0)
          {   // if so move it to the less than or equal portion
              j++;
              swapReferences(list, i, j);
          }
      }

      //restore pivot to correct spot
      swapReferences(list, start, j);
      quicksort( list, start, j - 1 );    // Sort small elements
      quicksort( list, j + 1, stop );   // Sort large elements
  }
```

Merge Sort Algorithm

Don Knuth cites John von Neumann as the creator of this algorithm

1. If a list has 1 element or 0 elements it is sorted
2. If a list has more than 2 split into into 2 separate lists
3. Perform this algorithm on each of those smaller lists
4. Take the 2 sorted lists and merge them together

Merge Sort

Ide metode Merge Sort :

1. Pembagian array data menjadi dua bagian (bagian kiri dan bagian kanan)

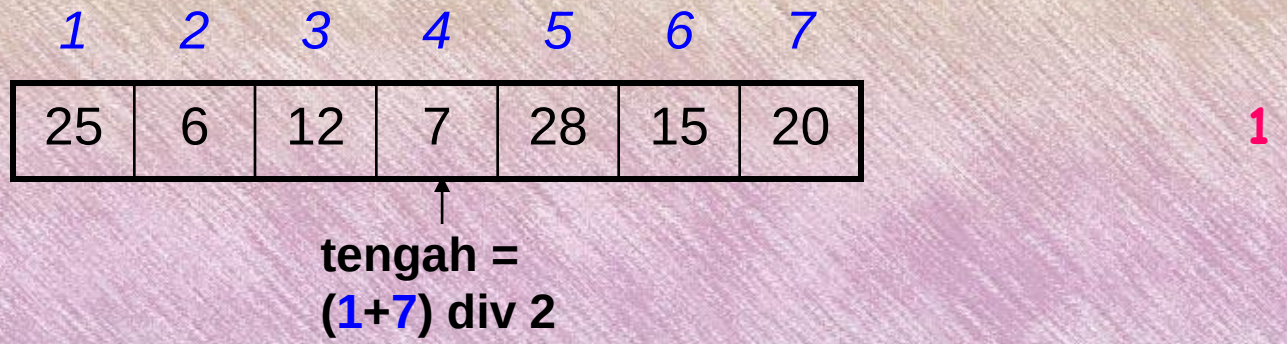
   Ulangi langkah 1 secara rekursi terhadap kedua bagian tersebut, sampai tiap subarray hanya terdiri dari satu elemen.

2. Gabungkan masing-masing bagian itu sekaligus mengurutkannya, sesuai pohon yang terbentuk saat membagi array, sampai membentuk array pertama saat sebelum dibagi.

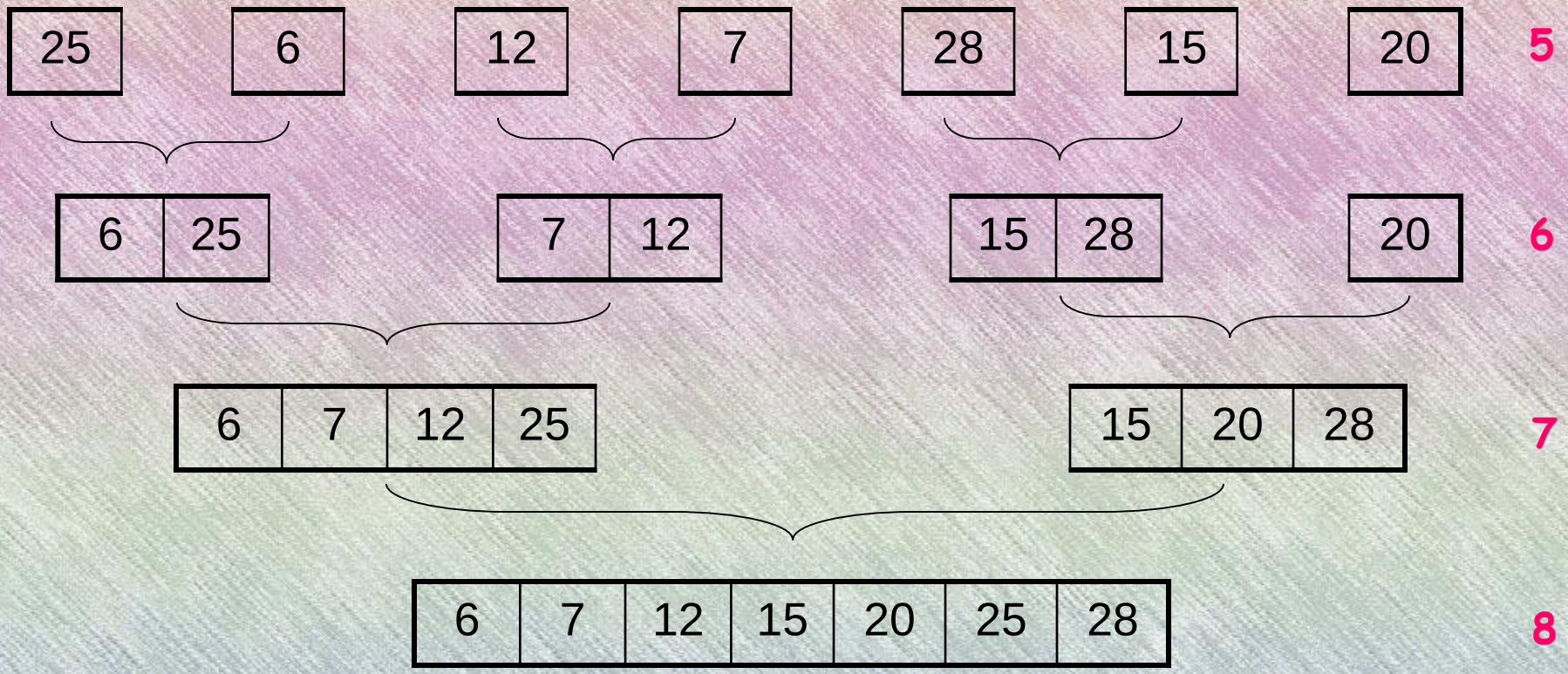
   Ulangi langkah 2 secara rekursi pula.

Ilustrasi Merge Sort

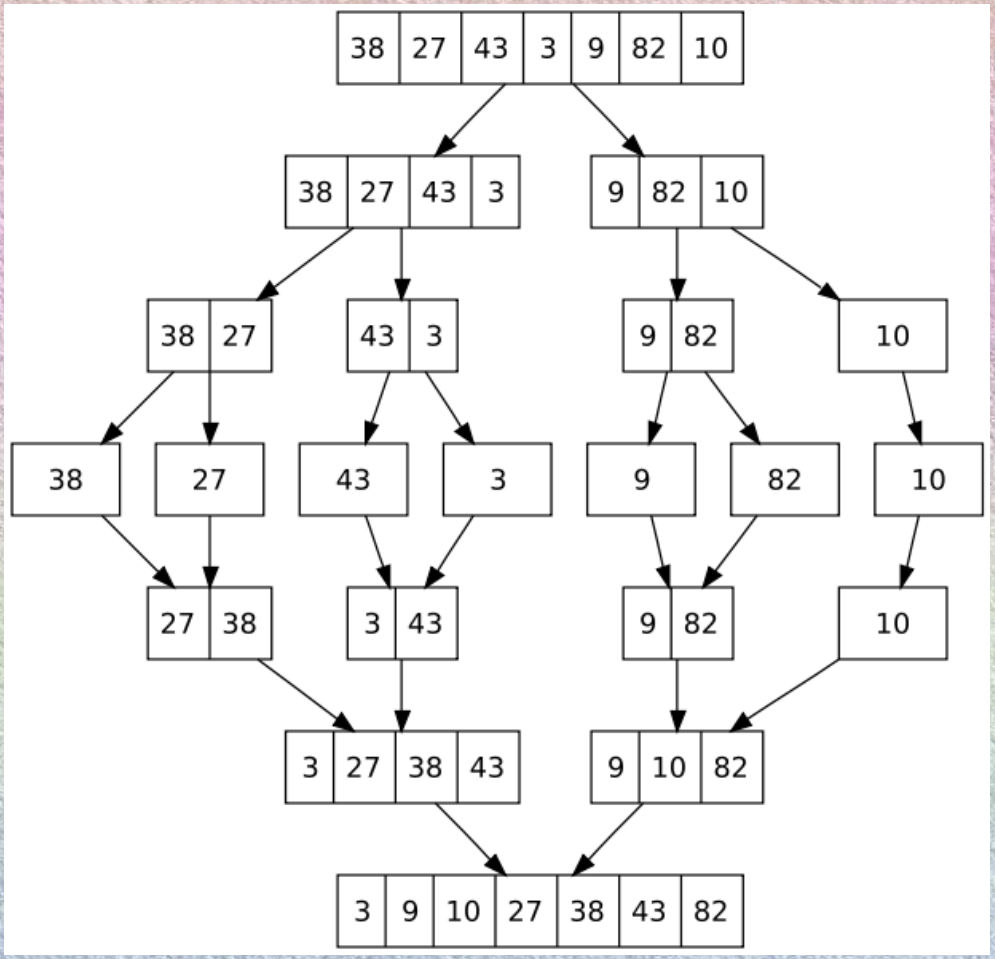
I



II


25 6 12 7 28 15 20
5
6 25 7 12 15 28 20
6
6 7 12 25 15 20 28
7
6 7 12 15 20 25 28
8

Merge Sort



When implementing one temporary array is used instead of multiple temporary arrays.

Why?

Algoritma MergeSortRekursi

```
Deklarasi
 Type TipeData : Array[1..100] of integer
 Data : TipeData
 n,i : integer
 Procedure MergeSort(input/output Data : TipaData;
                     input awal,akhir : integer)
Deskripsi
  output('Banyaknya elemen array :')
  input(n)
  i traversal[1..n]
     Data_i ← random(n)

  output('Data yang belum terurut : ')
  i traversal[1..n]
     output(Data_i)

  MergeSort(Data,1,n)

  output('Data yang sudah terurut :')
  i traversal[1..n]
     output(Data_i)
```

```
Procedure MergeSort(input/output Data : TipaData;
                    input awal,akhir : integer)
Deklarasi Lokal
  tengah : integer
  Procedure Merge(input/output Data : TipaData;
                  input awalkiri,akhirkiri,
                  awalkiri,akhirkiri : integer)
Deskripsi
  if (awal<akhir) then
      tengah ← (awal+akhir) div 2
      MergeSort(Data,awal,tengah)
      MergeSort(Data,tengah+1,akhir)
      Merge(Data,awal,tengah,tengah+1,akhir)
  endif
```

```
Procedure Merge(input/output Data : TipaData;
                input awalkiri,akhirkiri,
                awalkiri,akhirkiri : integer)
Deklarasi Lokal
  temp : TipeData
  i,kiri,kanan : integer
Deskripsi
  kiri ← awalkiri
  kanan ← awalkanan
  i ← awalkiri
  while (kiri<=akhirkiri) or (kanan<=akhirkanan) do
   if (Data_kiri<=Data_kanan) or (kanan>akhirkanan) then
      temp_i ← Data_kiri
      kiri ← kiri+1
   endif
   if (Data_kiri>Data_kanan) or (kiri>akhirkiri) then
      tempi ← Datakanan
      kanan ← kanan+1
   endif
   i ← i + 1
  endwhile
  i traversal[awalkiri..akhirkanan]
    Data_i ← temp_i
```

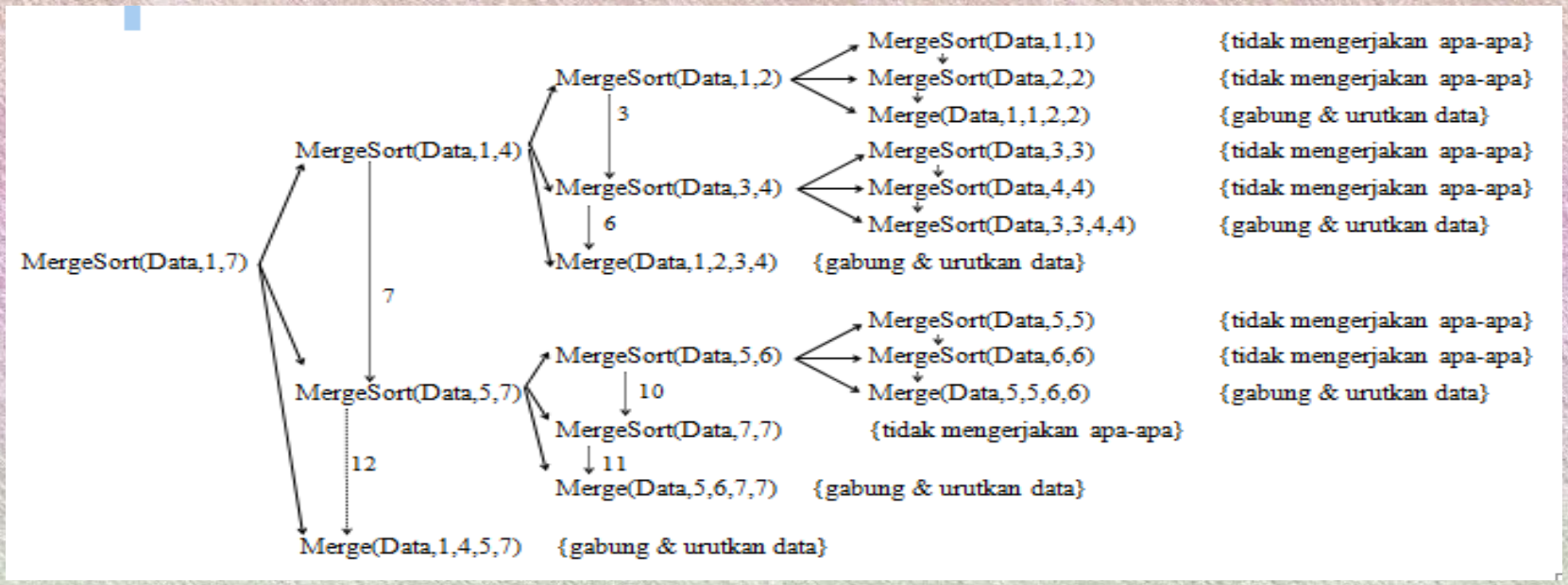
Merge Sort code

```
/**
 * perform a merge sort on the data in c
 * @param c c != null, all elements of c
 * are the same data type
 */
void mergeSort(Comparable[] c)
{       Comparable[] temp = new Comparable[ c.length ];
        sort(c, temp, 0, c.length - 1);
}

void sort(Comparable[] list, Comparable[] temp,
                        int low, int high)
{       if( low < high){
                int center = (low + high) / 2;
                sort(list, temp, low, center);
                sort(list, temp, center + 1, high);
                merge(list, temp, low, center + 1, high);
        }
}
```

Merge Sort Code

```
void merge( Comparable[] list, Comparable[] temp,
                        int leftPos, int rightPos, int rightEnd){
        int leftEnd = rightPos - 1;
        int tempPos = leftPos;
        int numElements = rightEnd - leftPos + 1;
        //main loop
        while( leftPos <= leftEnd && rightPos <= rightEnd){
                if( list[ leftPos ].compareTo(list[rightPos]) <= 0){
                        temp[ tempPos ] = list[ leftPos ];
                        leftPos++;
                }
                else{
                        temp[ tempPos ] = list[ rightPos ];
                        rightPos++;
                }
                tempPos++;
        }
        //copy rest of left half
        while( leftPos <= leftEnd){
                temp[ tempPos ] = list[ leftPos ];
                tempPos++;
                leftPos++;
        }
        //copy rest of right half
        while( rightPos <= rightEnd){
                temp[ tempPos ] = list[ rightPos ];
                tempPos++;
                rightPos++;
        }
        //Copy temp back into list
        for(int i = 0; i < numElements; i++, rightEnd--)
                list[ rightEnd ] = temp[ rightEnd ];
}
```

MergeSort bila ditest dengan data dari ilustrasi



POINTERS

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

Pointers

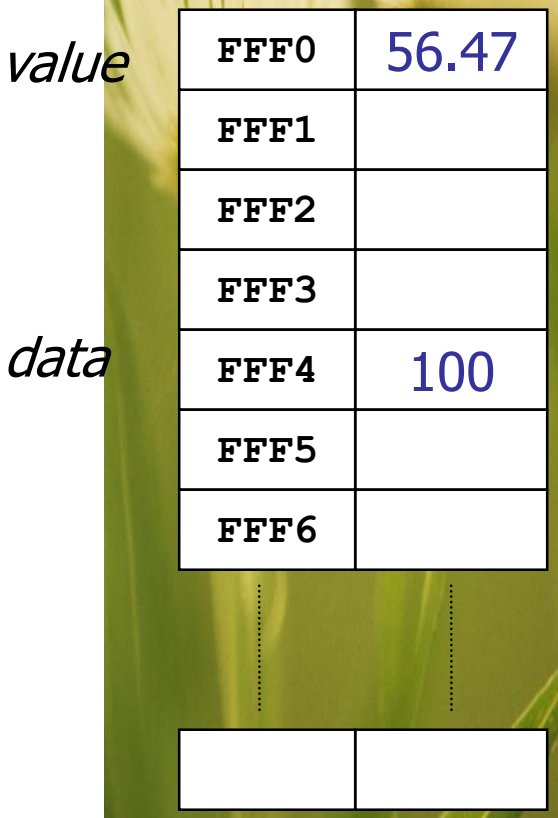
- Powerful feature of the C++ language
- One of the most difficult to master
- Essential for construction of interesting data structures

Addresses and Pointers

- C++ allows two ways of accessing variables
  - Name (C++ keeps track of the address of the first location allocated to the variable)
  - Address/Pointer
- Symbol **`&`** gets the address of the variable that follows it
- Addresses/Pointers can be displayed by the **`cout`** statement
  - Addresses displayed in HEXADECIMAL

Example

```
#include <iostream.h>

void main( )
{
    int data = 100;
    float value = 56.47;
    cout << data << &data << endl;
    cout << value << &value << endl;
}
```



Output:

```
100 FFF4
56.47 FFF0
```

Pointer Variables

- The pointer data type
  - A data type for containing an address rather than a data value
  - Integral, similar to **`int`**
  - Size is the number of bytes in which the target computer stores a memory address
  - Provides indirect access to values

Declaration of Pointer Variables

- A pointer variable is declared by:

```
dataType *pointerVarName;
```

  - The pointer variable *pointerVarName* is used to point to a value of type *dataType*
  - The * before the *pointerVarName* indicates that this is a pointer variable, not a regular variable
  - The * is not a part of the pointer variable name

Declaration of Pointer Variables (Cont ..)

- Example

```
int *ptr1;
float *ptr2;
```

  - **`ptr1`** is a pointer to an **`int`** value i.e., it can have the address of the memory location (or the first of more than one memory locations) allocated to an **`int`** value
  - **`ptr2`** is a pointer to a **`float`** value i.e., it can have the address of the memory location (or the first of more than one memory locations) allocated to a **`float`** value

Declaration of Pointer Variables (Cont ..)

- Whitespace doesn't matter and each of the following will declare **`ptr`** as a pointer (to a **`float`**) variable and **`data`** as a **`float`** variable

```
float *ptr, data;
float* ptr, data;
float (*ptr), data;
float data, *ptr;
```

Assignment of Pointer Variables

A pointer variable has to be assigned a valid memory address before it can be used in the program

Example:

```
float data = 50.8;
float *ptr;
ptr = &data;
```

This will assign the address of the memory location allocated for the floating point variable **`data`** to the pointer variable **`ptr`**. This is OK, since the variable **`data`** has already been allocated some memory space having a valid address

Assignment of Pointer Variables (Cont ..)

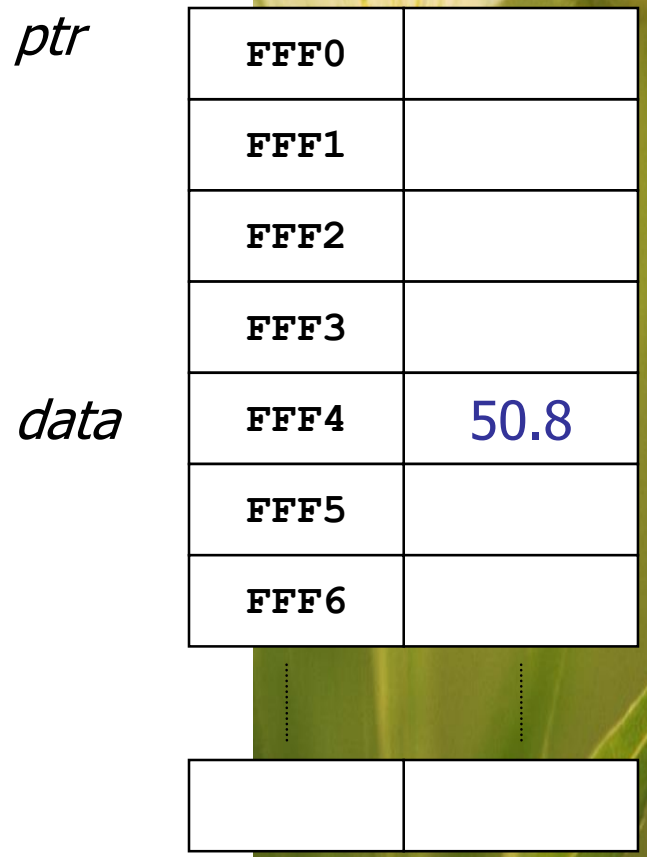
```
float data = 50.8;
float *ptr;
ptr = &data;
```

data

FFF0	
FFF1	
FFF2	
FFF3	
FFF4	50.8
FFF5	
FFF6	

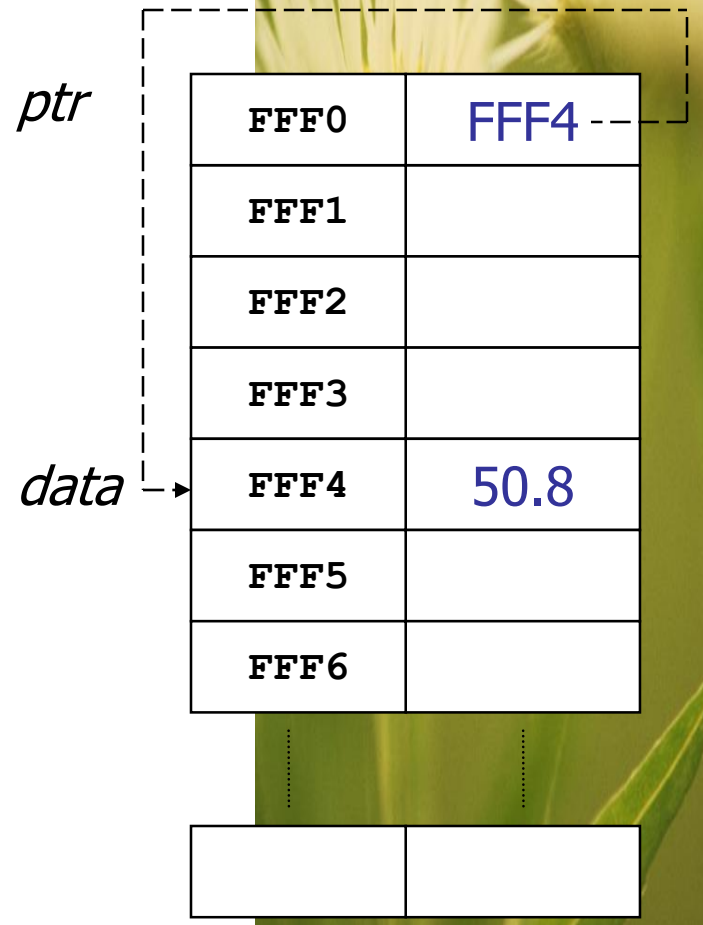
Assignment of Pointer Variables (Cont ..)

```
float data = 50.8;
float *ptr;
ptr = &data;
```



Assignment of Pointer Variables (Cont ..)

```
float data = 50.8;
float *ptr;
ptr = &data;
```



Assignment of Pointer Variables (Cont ..)

- Don’t try to assign a specific integer value to a pointer variable since it can be disastrous

```
float *ptr;
ptr = 120;
```

- You cannot assign the address of one type of variable to a pointer variable of another type even though they are both integrals

```
int data = 50;
float *ptr;
ptr = &data;
```

Initializing pointers

- A pointer can be initialized during declaration by assigning it the address of an existing variable

```
float data = 50.8;
float *ptr = &data;
```

- If a pointer is not initialized during declaration, it is wise to give it a **NULL** (0) value

```
int *ip = 0;
float *fp = NULL;
```

The `NULL` pointer

- The **`NULL`** pointer is a valid address for any data type.
  - But **`NULL`** is not memory address 0.
- It is an error to dereference a pointer whose value is **`NULL`**.
  - Such an error may cause your program to crash, or behave erratically.
  - It is the programmer's job to check for this.

Dereferencing

- *Dereferencing* – Using a pointer variable to access the value stored at the location pointed by the variable
  - Provide indirect access to values and also called *indirection*
- Done by using the *dereferencing operator* `*` in front of a pointer variable
  - Unary operator
  - Highest precedence

Dereferencing (Cont ..)

- Example:

```
float data = 50.8;
float *ptr;
ptr = &data;
cout << *ptr;
```

- Once the pointer variable **`ptr`** has been declared, **`*ptr`** represents the value pointed to by **`ptr`** (or the value located at the address specified by **`ptr`**) and may be treated like any other variable of **`float`** type

Dereferencing (Cont ..)

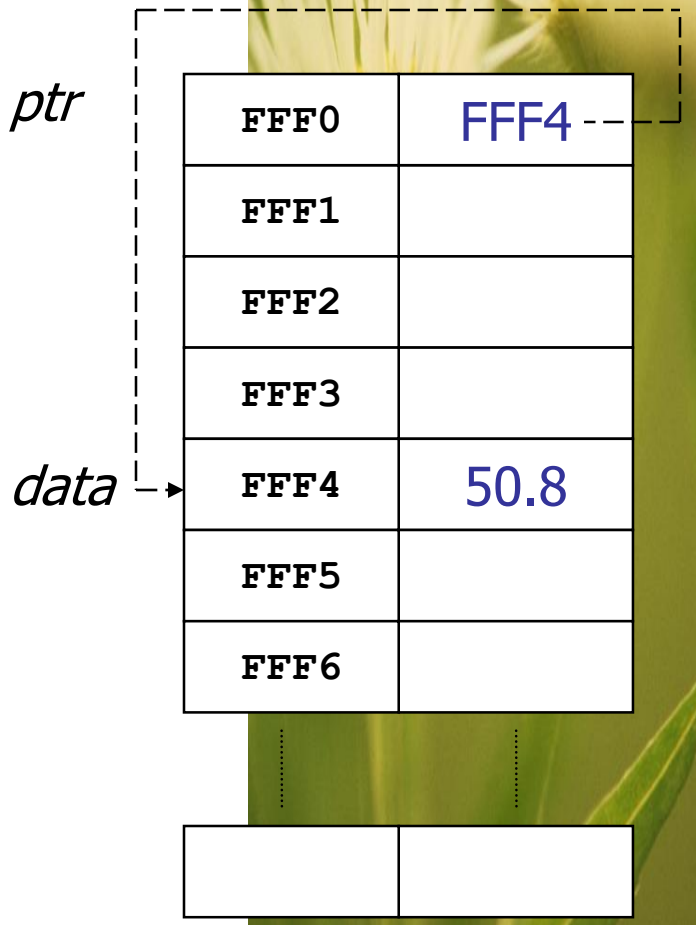
- The dereferencing operator **`*`** can also be used in assignments.

```
*ptr = 200;
```

  - Make sure that **`ptr`** has been properly initialized

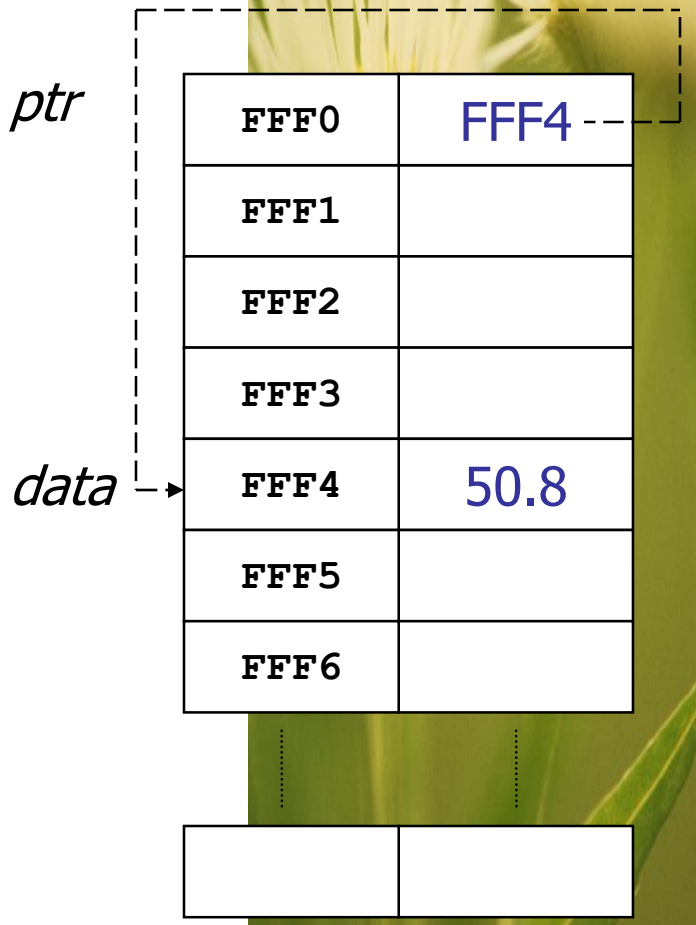
Dereferencing Example

```
#include <iostream.h>

void main()
{
    float data = 50.8;
    float *ptr;
    ptr = &data;
    cout << ptr << *ptr << endl;
    *ptr = 27.4;
    cout << *ptr << endl;
    cout << data << endl;
}
```



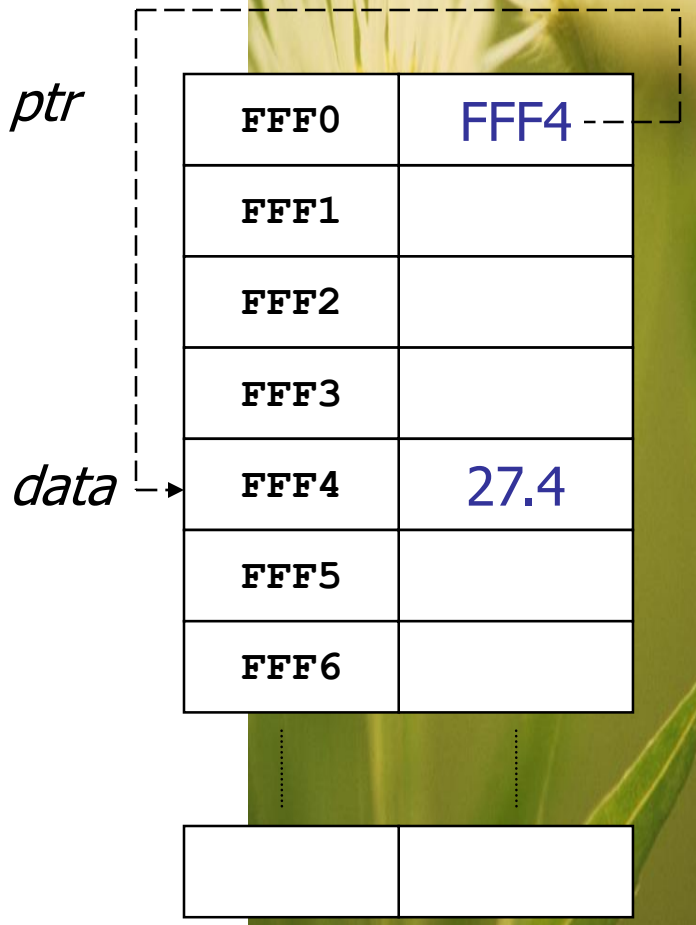
Dereferencing Example (Cont ..)

```
#include <iostream.h>

void main()
{
    float data = 50.8;
    float *ptr;
    ptr = &data;
    cout << ptr << *ptr << endl;
    *ptr = 27.4;
    cout << *ptr << endl;
    cout << data << endl;
}
```



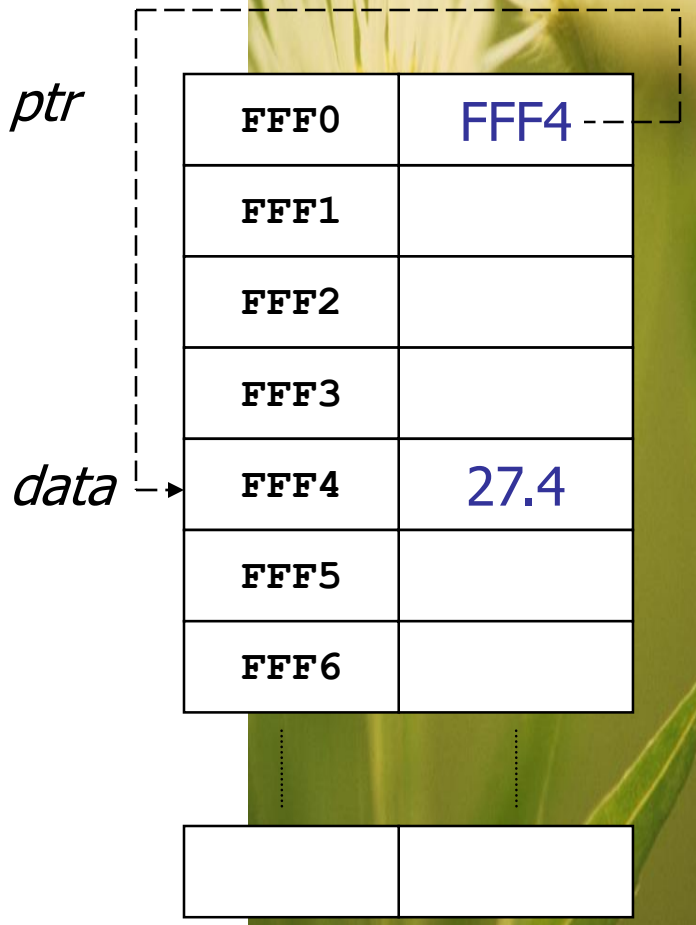
Dereferencing Example (Cont ..)

```
#include <iostream.h>

void main()
{
    float data = 50.8;
    float *ptr;
    ptr = &data;
    cout << ptr << *ptr << endl;
    *ptr = 27.4;
    cout << *ptr << endl;
    cout << data << endl;
}
```



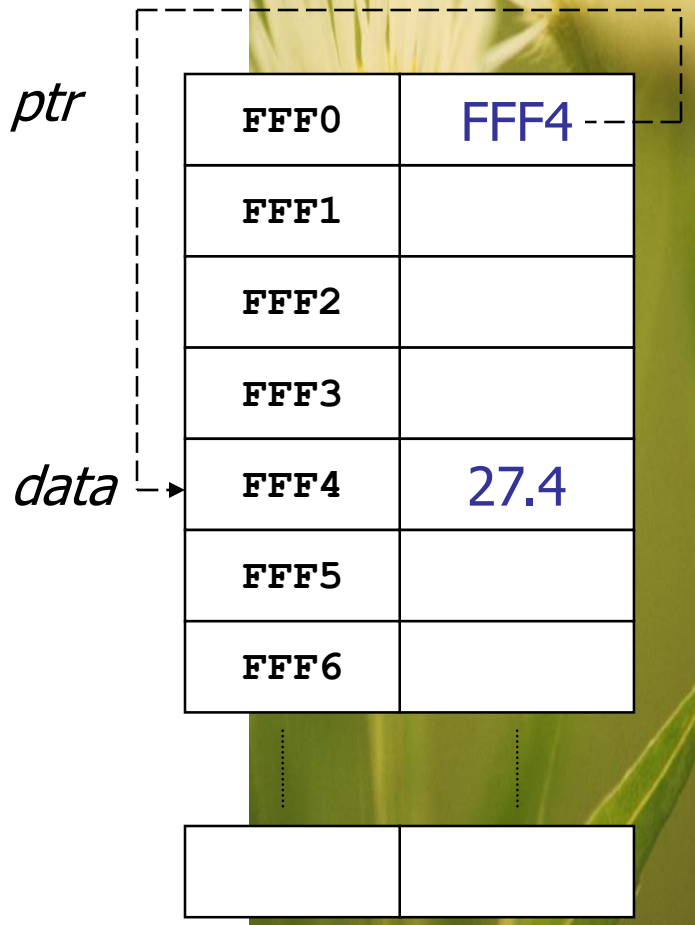
Dereferencing Example (Cont ..)

```
#include <iostream.h>

void main()
{
    float data = 50.8;
    float *ptr;
    ptr = &data;
    cout << ptr << *ptr << endl;
    *ptr = 27.4;
    cout << *ptr << endl;
    cout << data << endl;
}
```



Dereferencing Example (Cont ..)

```
#include <iostream.h>

void main()
{
    float data = 50.8;
    float *ptr;
    ptr = &data;
    cout << ptr << *ptr << endl;
    *ptr = 27.4;
    cout << *ptr << endl;
    cout << data << endl;
}
```



Operations on Pointer Variables

- Assignment – the value of one pointer variable can be assigned to another pointer variable of the same type
- Relational operations - two pointer variables of the same type can be compared for equality, and so on
- Some limited arithmetic operations
  - integer values can be added to and subtracted from a pointer variable
  - value of one pointer variable can be subtracted from another pointer variable

Pointers to arrays

- A pointer variable can be used to access the elements of an array of the same type.

```
int gradeList[8] =
{92,85,75,88,79,54,34,96};
int *myGrades = gradeList;
cout << gradeList[1];
cout << *myGrades;
cout << *(myGrades + 2);
cout << myGrades[3];
```

- Note that the array name **`gradeList`** acts like the pointer variable **`myGrades`**.

Pointers to arrays

- Sample program pointers to arrays

```
#include<iostream.h>
main()
{
  int numbers[5];
  int *p;

  p = numbers; *p = 10;
  p++; *p = 20;
  p = &numbers[2]; *p = 30;
  p = numbers + 3; *p = 40;
  p = numbers; *(p+4) = 50;
  for(int n=0;n<5;n++)
    cout << numbers[n] << ", ";
 }
```

Output :

```
10, 20, 30, 40, 50,
```

Declaration of pointers

Pseodocode :

name_pointer : pointer to tipedata

C++ :

```
tipedata  *name_pointer;
```

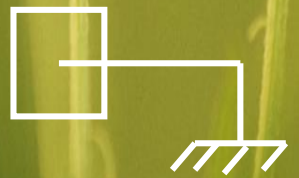
(*pointer null*)

```
name_pointer =
     (tipedata *) malloc(size_t size);
```

(*empty pointer*)

Illustration:

name_pointer



name_pointer

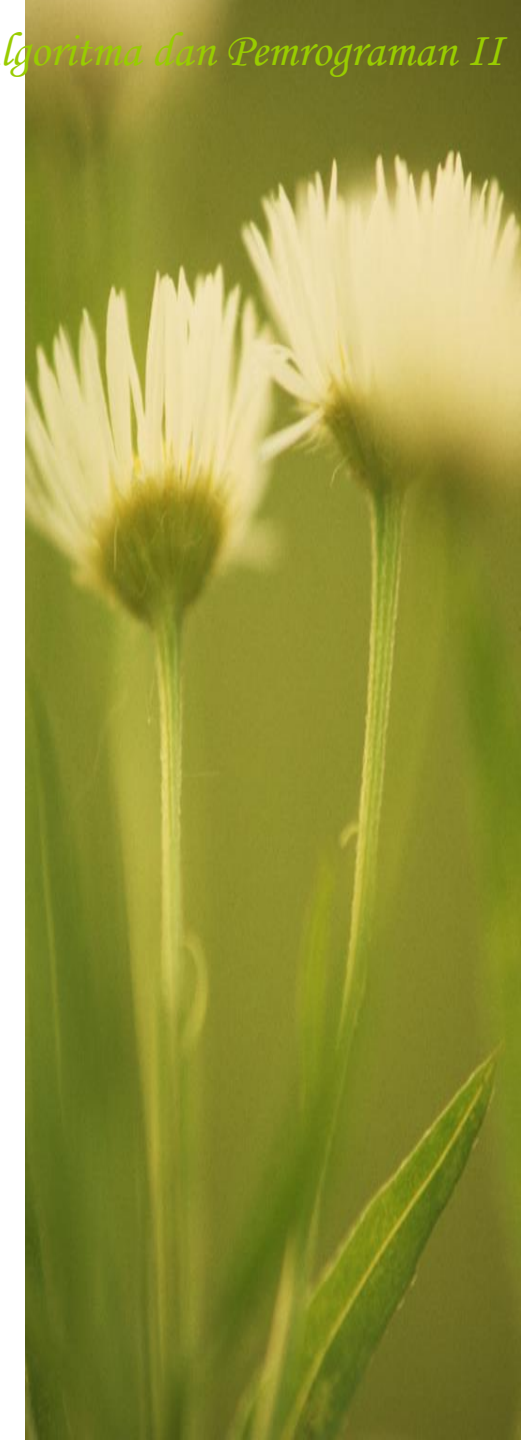
Example

Pseodocode :

p : pointer to integer
nilai : pointer to real
s : pointer to char

C++ :

```
int *p;
float *nilai;
char *s;
```

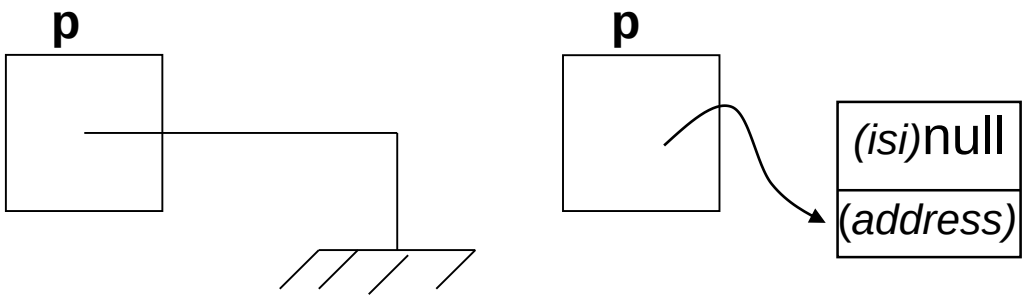


The declaration of a null pointer in memory in C ++

***malloc(size_t size)**

- Example:

```
int *p; int *q;
p=(int *) malloc(sizeof(int));
q=(int *) malloc(sizeof(int));
```

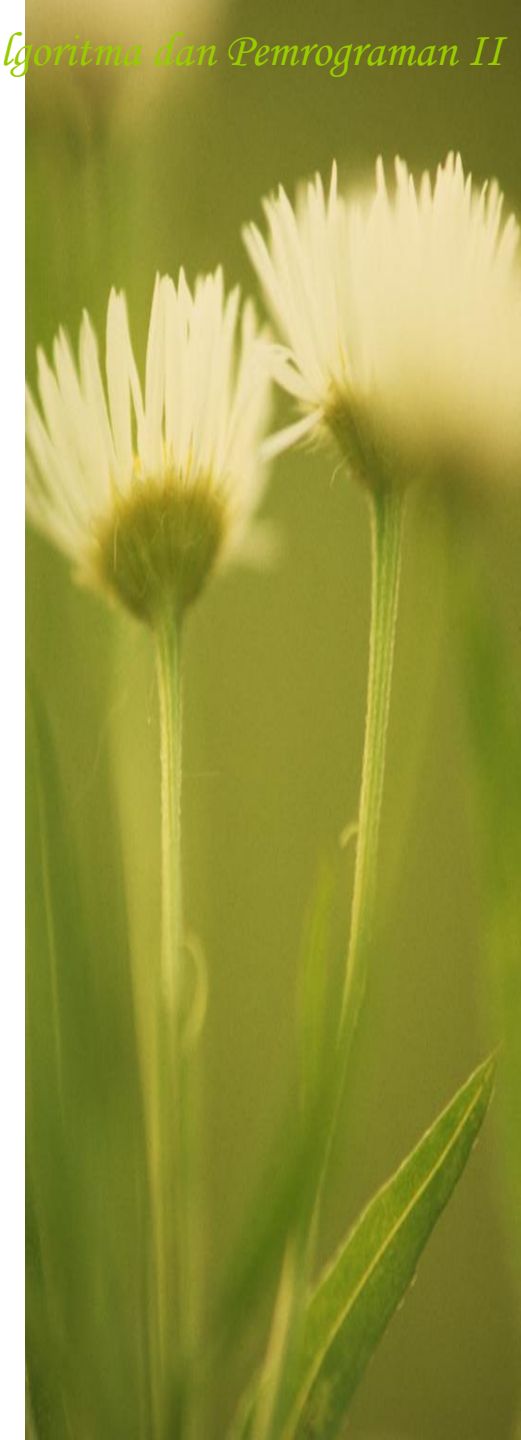


Accessing with pointer

- to access the value / content on the memory pointed to by the pointer used the symbol '*'
- Example :

  *p = 10;

  *q = 20;



Memory pointer designated another pointer

- Example :

```
p = q;
```

- means p points to the memory address designated by q , and thus p and q designate the same memory address.

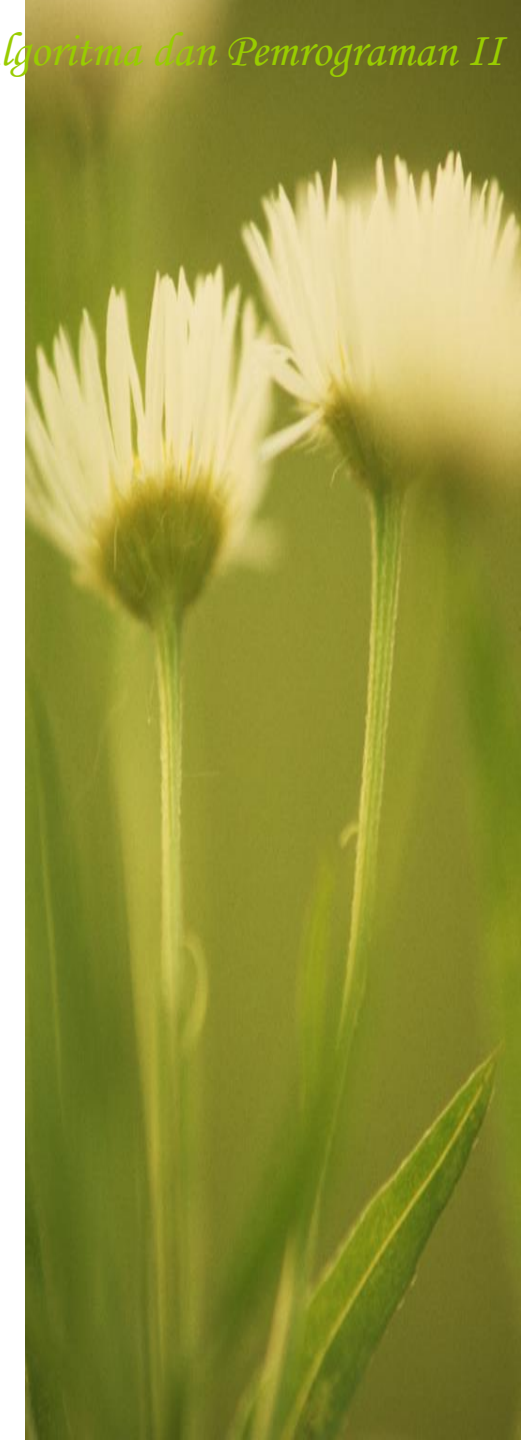
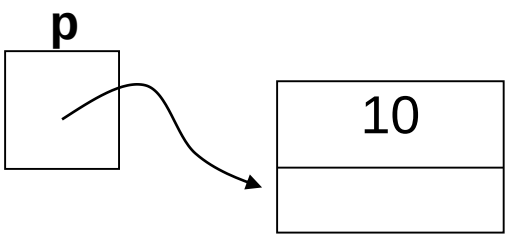
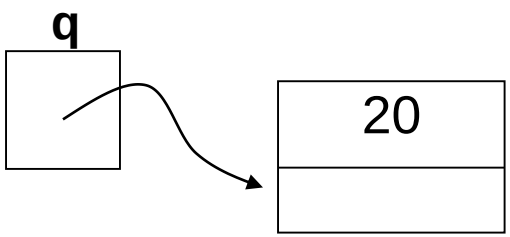


illustration :

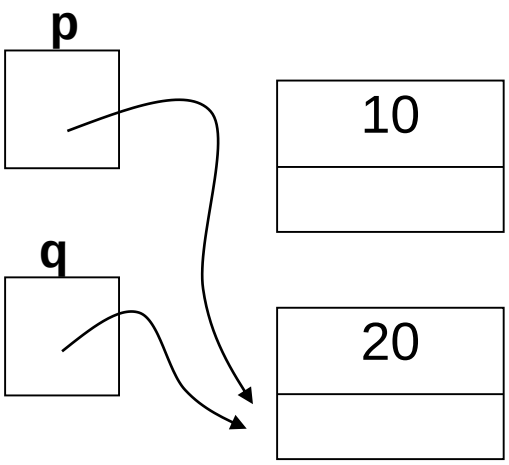
***p = 10**



***q = 20**



p = q



//sample program Pointer1.cpp

```
#include <iostream.h>
#include <alloc.h>
#include <stdlib.h>
void main()
{
    int  *p, *q;
    p=(int *)malloc(sizeof(int));
    q=(int *)malloc(sizeof(int));
    *p=10;
    *q=20;
    cout<<"Isi info pointer :\n";
    cout<<"*p = "<<*p<<endl;
    cout<<"*q = "<<*q<<endl;
    cout<<"\nAlamat register pointer :\n";
    cout<<"p = "<<p<<endl;
    cout<<"q = "<<q<<endl;
    p=q;
    cout<<"\nKondisi akhir isi info pointer :\n";
    cout<<"*p = "<<*p<<endl;
    cout<<"*q = "<<*q<<endl;
}
```

Output :

```
Isi info pointer :
*p = 10
*q = 20

Alamat register pointer :
p = 0x212f01d2
q = 0x212f01ca

Kondisi akhir isi info pointer :
*p = 20
*q = 20
```

Pointers pointing static variables

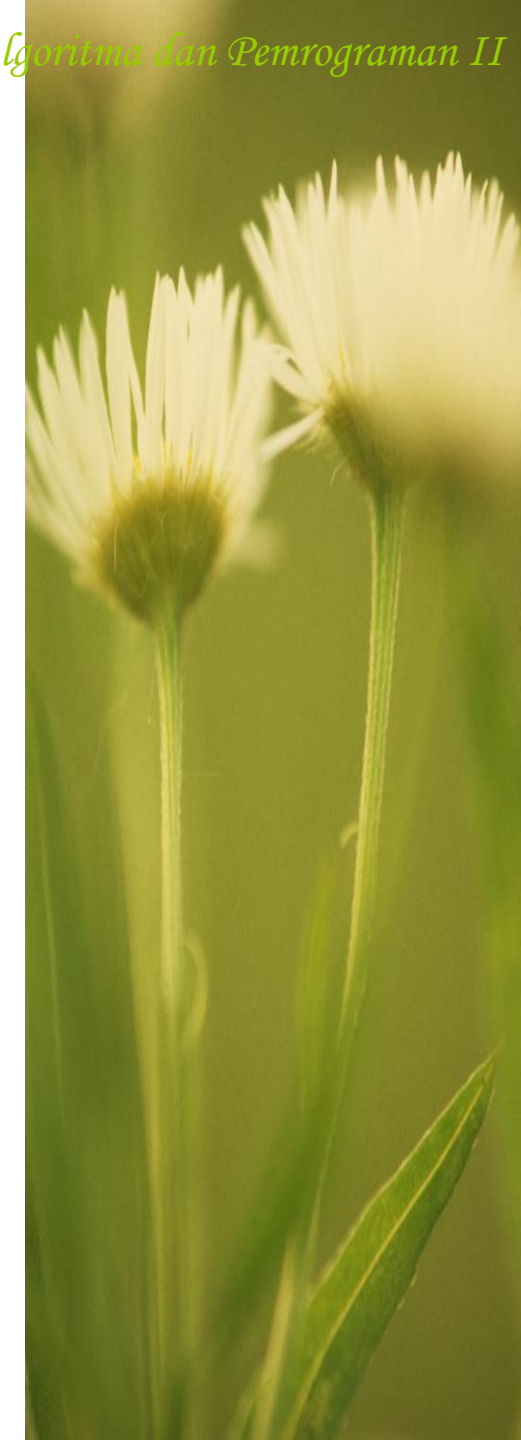
Suppose that px is a variable of type pointer that will contain the address of another variable of type integer , then declared

```
int x;
int *px;
```

To set the pointer to point to other variables, first pointer to be filled with the address of the variable to be appointed.

Operator ' & ' is used to declare a static variable address to be designated .
example :

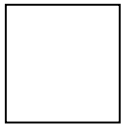
```
px = &x;
```

illustration

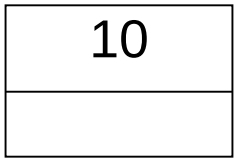
int *px;

px



int x=10;

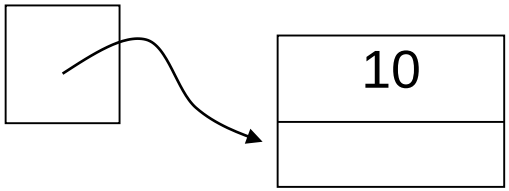
x



px = &x

px

x



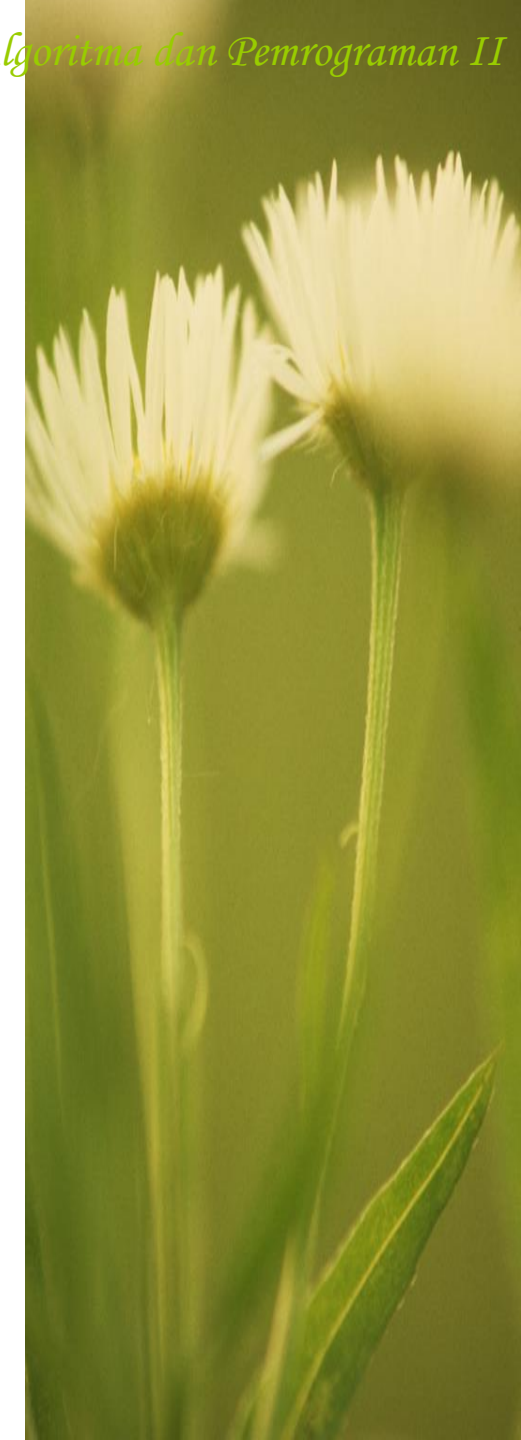
If a static variable has been designated by the pointer, the contents of these variables can be accessed via the variable itself (direct access) or via a pointer (indirect access).

direct access is done directly by its static variables (not a pointer). Example :

```
x = 10
```

Indirection operator (indirect access), in the form of the symbol ' * '.
example :

```
*px = 10
```



//Sample program Pointer3.cpp

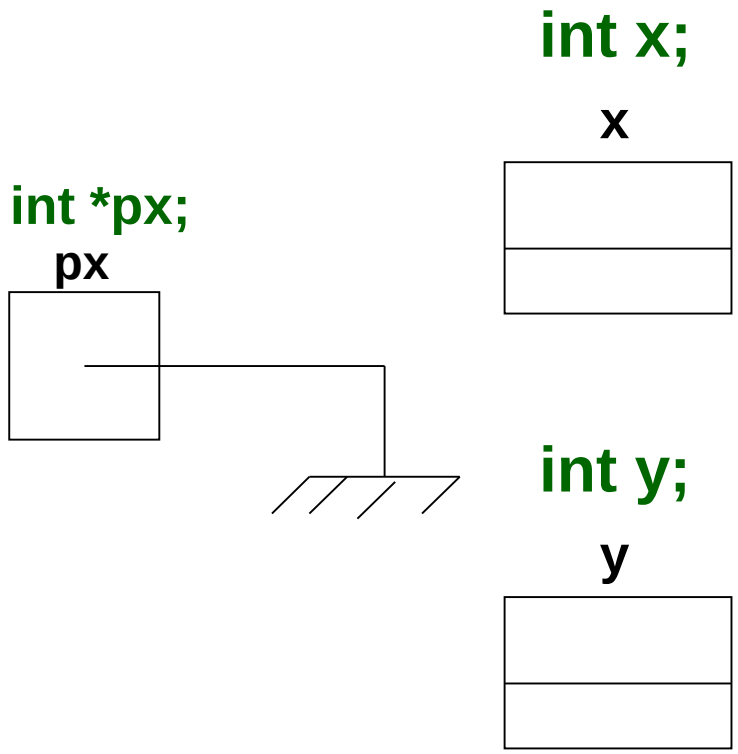
```
#include <iostream.h>
main()
{
   int x,y;   //x dan y bertipe int
   int *px;   //deklarasi px, pointer yang menunjuk obyek
              //bertipe int
a  x=87;
b  px=&x;     //px berisi alamat dari x
c  y =*px;    //y berisi nilai yang ditunjuk px

   cout<<"alamat x= "<<&x<<endl;
   cout<<"nilai  x= "<<x<<endl;
   cout<<"alamat yang ditunjuk oleh px= "<<px<<endl;
   cout<<"nilai yang ditunjuk oleh  px= "<<*px<<endl;
   cout<<"alamat y= "<<&y<<endl;
   cout<<"nilai  y= "<<y;
}
```

Output :

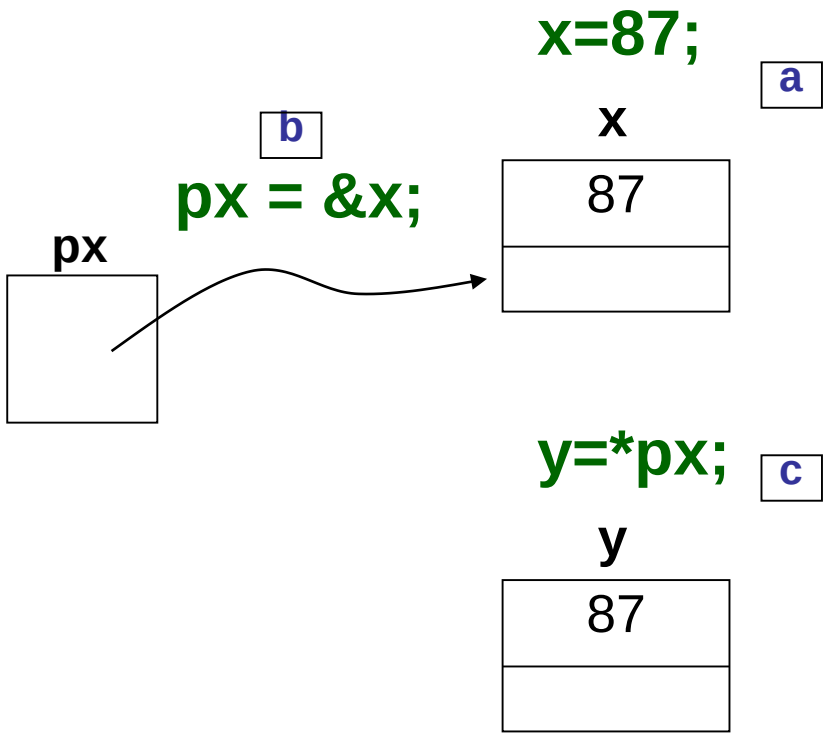
```
alamat x= 0x301f2452
nilai  x= 87
alamat yang ditunjuk oleh px= 0x301f2452
nilai yang ditunjuk oleh  px= 87
alamat y= 0x301f2450
nilai  y= 87
```

Illustration pointer3.cpp

1



2

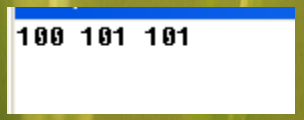


Operator Pointer

Sample Program :

```
#include<iostream.h>
main()
{
  int *ptr, num;
  ptr = &num;
  *ptr = 100;
  cout << num << " ";
  (*ptr)++;
  cout << num << " ";
 *ptr= (*ptr)*2;
  cout << num << "\n";
}
```

Output :



Expresi Pointer

- Only four arithmetic operators can be used on a pointer: ++, -- , + , dan –
- Sample program pointer Aritmatika.

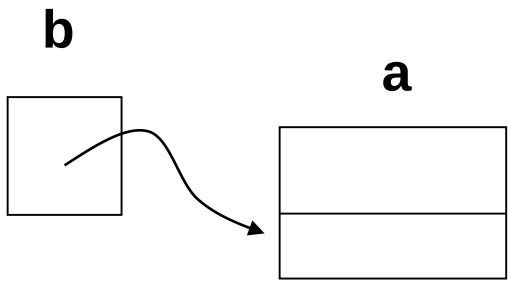
```
#include<iostream.h>
main()
{ int i[10], *i_ptr;
  double f[10], *f_ptr;
  int x;
  i_ptr = i;          //i_ptr points to first element of i
  f_ptr = f;          //f_ptr points to first element of f
  for(x=0;x<10;x++)
      cout << i_ptr+x << " " << f_ptr+x << "\n";
}
```

Output :

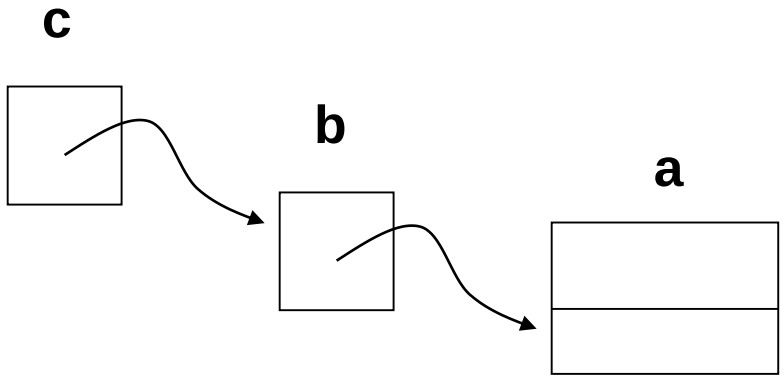
```
0x197f25e0 0x197f2590
0x197f25e2 0x197f2598
0x197f25e4 0x197f25a0
0x197f25e6 0x197f25a8
0x197f25e8 0x197f25b0
0x197f25ea 0x197f25b8
0x197f25ec 0x197f25c0
0x197f25ee 0x197f25c8
0x197f25f0 0x197f25d0
0x197f25f2 0x197f25d8
```

Pointers to Pointers

- *Pointer Single*



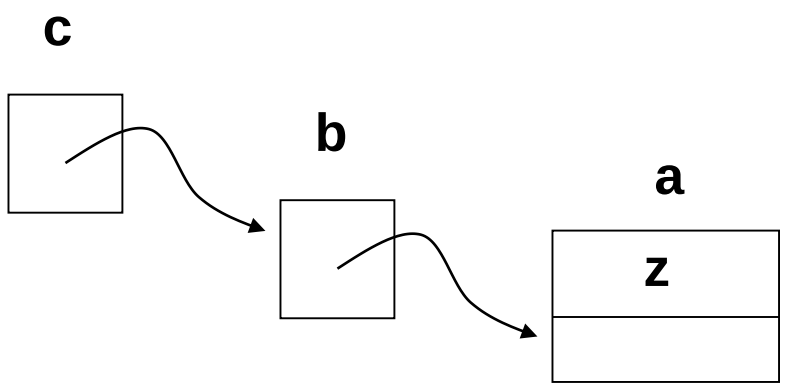
- *Pointers to Pointers*



Pointer to Pointer

- Example :

```
char a;
char *b;
char **c;
a = ‘z’;
b = &a;
c = &b;
```



- Sample program 5 (pointer to pointer)

```
#include<iostream.h>
main()
{
  int x, *p, **q;
  x = 10;
  p = &x;
  q = &p;
  cout << *p << endl;  //prints the value of x
  cout << **q;              //prints the value of x
}
```

Output :

Output ?

```
#include<iostream.h>
main()
{
  int m, n, *x, **y;
  m = 10;
  x = &m;
  y = &x;
  n = **y + 5;
  x = &n;
  cout << m << endl;
  cout << n << endl;
  cout << *x << endl;
  cout << **y << endl;
 }
```


Output ?

```
#include<iostream.h>
main()
{
  int x, *p, **q;
  x = 10;
  p = &x;
  q = &p;
  cout << "x = " << x << endl;
  cout << "*p = " << *p << endl;
  cout << "**q = " << **q << endl;
  cout << "&x = " << &x << endl;
  cout << "&p = " << &p << endl;
  cout << "&q = " << &q << endl;
  cout << "p = " << p << endl;
  cout << "q = " << q << endl;
  cout << "*q = " << *q << endl;
}
```

Output :

```
x   = 10
*p  = 10
**q= 10
&x  =  0x21ef241e
&p  =  0x21ef241a
&q  =  0x21ef2416
p   =  0x21ef241e
q   =  0x21ef241a
*q  =  0x21ef241e
```

- Distinguish !

File (1)

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

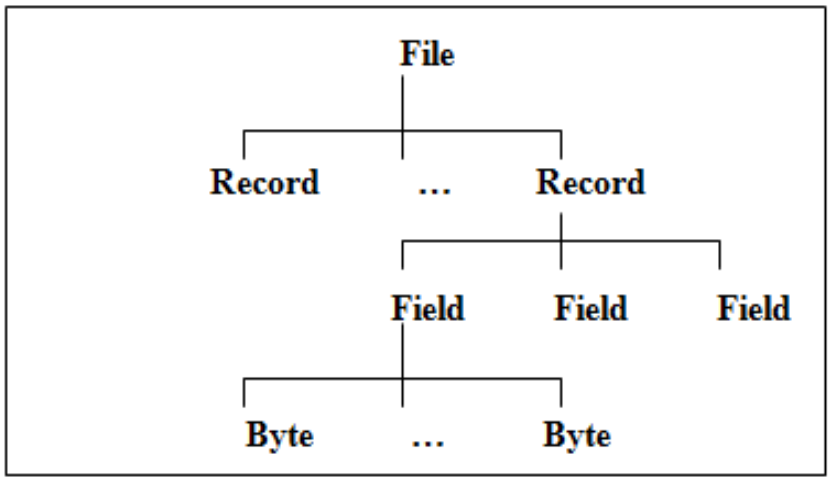
File (1)

Storage :

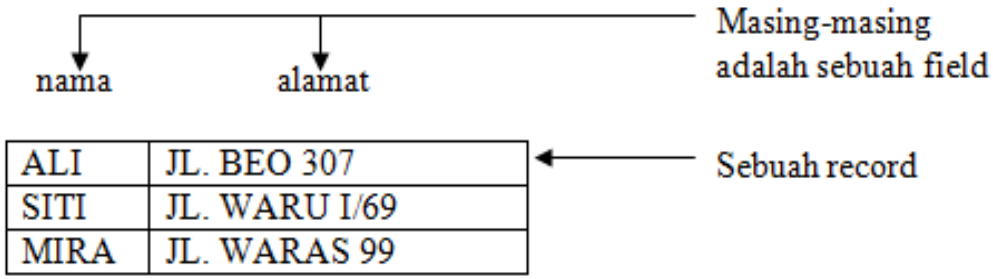
- Main Memory (RAM) → not store information permanently
- Secondary Memory →permanent storage media . Example : disk (floppy disk , hard disk , compact disk , flash disk etc .) . The data stored in the secondary storage are grouped in the form of records / files

File Structur:

- A file is the organization of a number of record may consist of one or several fields , and each field consists of one or more bytes. The byte is the arrangement of 8 bits .



Struktur data dari file



Gambaran record dan field

Organizing the data in the file :

•A so-called datum or record information , while the plural is called data. All the records in a file organized storage , and accessing the records in a file depends on the method perorganisasiannya TSB.

•There are two ways : successive (sequential) and random (random).

Archive streak

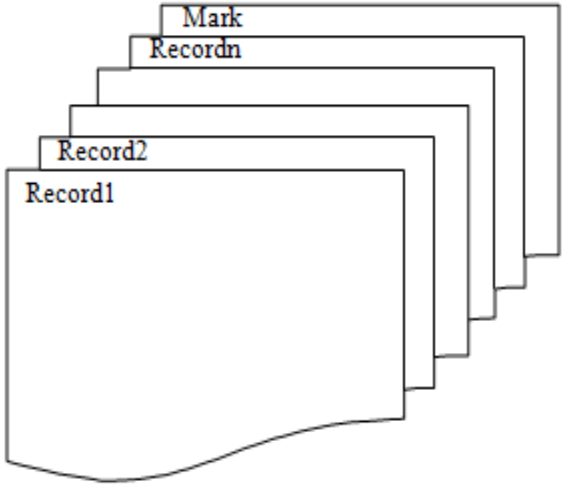
Understanding streak Archive : A collection of integrated records , which are stored in the secondary storage , which can be accessed sequentially record per record in the same direction starting from the first record.

Files can be viewed in two ways :

1. Secara Tabel

	Field1	Field2	Field3	…	Fieldn
Record1					
Record2					
Record3					
…					
Recordn					
Mark					

2. Secara Blanko



Record terakhir adalah record fiktif yang menandai akhir dari file.

Deklaration File
(pseodocode)

Type NamaRecord : TipeRecord
NamaArsip : SeqFile of TipeRecord
VarRecord : NamaRecord

Example :

```
Type DataMhs : Record
                    <NIM : Integer,
                     Nama : String,
                IPK : Real >
ArsipMhs : SeqFile of DataMhs
Mhs : DataMhs

Type ArsipBil :SeqFile of Integer
Bil : ArsipBil
i : integer

Type ArsipKar : SeqFile of Char
Kar : ArsipKar
c : Char
```

C++ :

typedef TipeRecord NamaRecord;

FILE *NamaArsip;

NamaRecord VarRecord;

Example :

```
Typedef struct {long NIM;
                char Nama[25];
                float IPK;
                } DataMhs;
FILE *ArsipMhs;
DataMhs Mhs;
```

```
FILE *Bil;
int i;
```

```
FILE *Kar;
char c;
```

Raw commands (pseodocode)

- **Open(NamaArsip,kode)**

  Ex :

  Open(ArsipMhs,1)

  Open(Bil,2)

  Open(Kar,1)

- **FRead(NamaArsip,VarRecord)**

  Ex :

  FRead(ArsipMhs,Mhs)

  FRead(Bil,i)

  FRead(Kar,c)

- **FWrite(NamaArsip,VarRecord)**

  Ex :

```
FWrite(ArsipMhs,<12331,'Hanif',3.50)
FWrite(ArsipMhs,<99999,'xxxxx',9.99)
FWrite(Bil,765)
FWrite(Kar,'R')
Input(i)
FWrite(Bil,i)
Input(c)
FWrite(Kar,c)
Input(Mhs.NIM)
Input(Mhs.Nama)
Input(Mhs.IPK)
FWrite(ArsipMhs,Mhs)
```

- **Close(NamaArsip)**

  Ex :

```
Close(ArsipMhs)
```

C++:

- NamaArsip1 = fopen(NamaArsipFisik1,"r"); /*utk dibaca*/
  NamaArsip2 = fopen(NamaArsipFisik1,"w"); /*utk ditulis*/
  Contoh :

```
ArsipMhs = fopen("C:ArsipMhs.dat","r");
Bil = fopen("A:Bil.dat","w");
Kar = fopen("A:Kar.dat","r");
```

- fread(VarRecord,sizeof(VarRecord),jum_record,NamaArsip);
  Contoh :

```
fread(&Mhs,sizeof(Mhs),1,ArsipMhs);
fread(&i,sizeof(i),1,Bil);
fread(&c,sizeof(c),1,Kar);
```

- fwrite(VarRecord,sizeof(VarRecord),jum_record,NamaArsip);
  Contoh :

```
fwrite(&Mhs,sizeof(Mhs),1,ArsipMhs);
fwrite(&i,sizeof(i),1,Bil);
fwrite(&c,sizeof(c),1,Kar);
fwrite(&Kar,'R')
```

- fclose(NamaArsip);
  Contoh :

```
fclose(ArsipMhs);
```

Sample Program 1:

```
// store integer data
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
FILE *Bil;
typedef enum {true=1,false=0} boolean;
boolean Mark(int i);

main()
{
  int i;

  Bil = fopen("Bil.dat","w");
  cout << "Bilangan : "; cin >> i;
  while (Mark(i) != true)
  {
                fwrite(&i,sizeof(i),1,Bil);
                cout << "Bilangan : "; cin >> i;
  }

i = 999;
  fwrite(&i,sizeof(i),1,Bil);
  fclose(Bil);

}

boolean Mark(int i)
{
     return (i == 999);
}
```

Sample Program 2 :

```
// read data integer is in the file of the sample program 1
#include <stdio.h>
#include <iostream.h>
#include <conio.h>

FILE *Bil;
typedef enum {true=1,false=0} boolean;
boolean Mark(int i);

main()
{
 int i,jum;
 jum=0;
 Bil = fopen("Bil.dat","r");
 fread(&i,sizeof(i),1,Bil);
 if (Mark(i))
      cout << "Arsip kosong\n";
 else
```

```
{
            do
            {   jum=jum+i;
                        //cout
<< i <<endl;

           fread(&i,sizeof(i)
,1,Bil);
            }
            while (!Mark(i));
  }
  cout <<jum;
  fclose(Bil);
  return 0;
}
boolean Mark(int i)
{
          return (i == 999);
}
```

Sample program 3 :

```
// program storing student data and student
//print data in accordance with the desired nim

#include <stdio.h>
#include <iostream.h>
#include <conio.h>
#include <string.h>
#include <iomanip.h>
  typedef struct { long NIM;
        char Nama[25];
        char KodeMK[5];
        int SKS;
        char Indeks;
      } DataMhs;
FILE *ArsipMhs;
typedef enum {true=1,false=0} boolean;
void RekamDataMahasiswa();
void CetakDaftarNilai(long KodeNIM);
boolean Mark(DataMhs Mhs);
```

```
main()
{
  long KodeNIM;

  RekamDataMahasiswa();
  clrscr();
  cout << " NIM : "; cin >> KodeNIM;
  clrscr();
  CetakDaftarNilai(KodeNIM);
  return 0;
}
 void RekamDataMahasiswa()
{
  DataMhs Mhs;
   ArsipMhs = fopen("Mhs.dat","w");
  cout << "NIM : "; cin >> Mhs.NIM;
  while (Mark(Mhs) != true)
  {
                cout << "Nama : "; cin >> Mhs.Nama;
                cout << "Kode MK : "; cin >>
Mhs.KodeMK;
                cout << "SKS : "; cin >> Mhs.SKS;
                cout << "Nilai : "; cin >> Mhs.Indeks;
                fwrite(&Mhs,sizeof(Mhs),1,ArsipMhs);
                cout << "NIM : "; cin >> Mhs.NIM;
```

```
  }
  Mhs.NIM = 9999;
  strcpy(Mhs.Nama, "xxxxx");
  strcpy(Mhs.KodeMK, "xxxxx");
  Mhs.SKS = 9;
  Mhs.Indeks = 'x';
  fwrite(&Mhs,sizeof(Mhs),1,ArsipMhs);
  fclose(ArsipMhs);
}

void CetakDaftarNilai(long KodeNIM)
{
  int no;
 DataMhs Mhs;
  boolean ketemu;
  boolean stop;
  ArsipMhs = fopen("Mhs.dat","r");
  stop = false;
  fread(&Mhs,sizeof(Mhs),1,ArsipMhs);
  if (Mark(Mhs))
    cout << "Arsip kosong\n";
  else
  { //cari code NIM
    ketemu = false;
    do
    {
              if (Mhs.NIM == KodeNIM)
                        ketemu = true;
              else
                        fread(&Mhs,sizeof(Mhs),1,ArsipMhs);
    }
    while (!ketemu && !Mark(Mhs));
    if (ketemu)
    {
```

```
cout << "Daftar Nilai Mata Kuliah\n";
                cout << " NIM   : " << Mhs.NIM << endl;
                cout << " Nama  : " << Mhs.Nama << endl;
                cout << "-------------------------------\n";
                cout << "No. Mata Kuliah    SKS    NIlai\n";
                cout << "-------------------------------\n";
                no = 0;
                do
                {
                                no++;
                                cout << no << setw(7) << Mhs.KodeMK;
            cout << setw(13) << Mhs.SKS << setw(9);
            cout << Mhs.Indeks << endl;
                                fread(&Mhs,sizeof(Mhs),1,ArsipMhs);
                }
                while (Mhs.NIM == KodeNIM && !Mark(Mhs));
                cout << "-------------------------------\n";
    }
    else
    {           cout << "Data mahasiswa dengan NIM = ";
         cout << KodeNIM << " tidak ada";
      }
    }
    fclose(ArsipMhs);
}

boolean Mark(DataMhs Mhs)
{
    return (Mhs.NIM == 9999);
}
```

File (2)

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

File Operation

Open / Enable File

```
FILE fopen(char *namafile, char *mode);
```

Description mode :

- **r** : Read only
- **w** : Menyatakan file baru diciptakan. Operasi yang akan dilakukan adalah operasii perekaman
- data. Jika file tersebut sudah ada, isi yang lama akan dihapus.
- **a** : Membuka file yang ada pada disk dan operasi yang akan dilakukan adalah operasii penambahan data pada file. Jika file belum ada, secara otomatis file akan dibuat .
- **r+** : Membuka file yang sudah ada, operasi yang dilakukan berupa pembacaan dan penulisan.
- **w+ :** Membuka file untuk pembacaan/penulisan. Jika file sudah ada, isinya akan dihapus.
- **a+** : Membuka file, operasi yang dilakukan berupa perekaman dan pembacaan. Jika file sudah ada, isinya tak akan terhapus.

- Binary files are files on the disk storage schemes are in binary form, ie as forms of memory (RAM) computer. While the text file is a file that the pattern of data storage in the form of characters.

- Description mode :

rt : mode is a text file and the file to be read.

rt+ : mode is a text file and the file can be read or write (= **r+t**).

rb : mode is a binary file and the file to be read.

- **Close File**

```
Int fclose(FILE *pf);
```

- **Saving File per Character**

```
Int fputc(int kar, FILE *ptr_file );
```

- **Read File per Character**

```
Int fgetc(FILE *ptr_file );
```

SAMPLE PROGRAM :

```
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
#define CTRL_Z 26
main()
{
  FILE *pf;
  char kar;
  clrscr();
  if((pf=fopen("COBA.TXT","w"))==NULL)
  {
     cout<<" File tak dapat diciptakan !!\r\n";
     exit(1);
  }
  while((kar=getche()) != CTRL_Z)
              fputc(kar,pf);
}
```

SAMPLE PROGRAM :

```
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
main()
{
  FILE *pf;
  char kar;
  clrscr();
  if((pf=fopen("COBA.TXT","r"))==NULL)
  {
     cout<<" File tak dapat dibuka !!\r\n";
     exit(1);
  }
  while((kar = getc(pf)) != EOF)
    cout<<kar;
  fclose(pf);
  getche();
}
```

Reading File per Integer

```
Int getw(FILE *ptr_file);
```

Saving File per Integer

```
Int putw(FILE *ptr_file);
```

SAMPLE PROGRAM :

```
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
main()
{
  FILE *pf;
  int nilai;
  char jawab;
 clrscr();
  if((pf=fopen("BILANGAN.DAT","wb"))==NULL)
            {
            cout<<"File gagal diciptain !!\n";
            exit(1);
    }
  cout<<"Program untuk menyimpan data integer ke file.";

  do
    {
            cout<<"\r\n  Bilangan yang akan disimpan : ";
            cin>>nilai;
            putw(nilai, pf);
            cout<<"  Memasukkan data lagi (Y/T)? : ";
            jawab =getche();
    }
  while(jawab == 'y' || jawab == 'Y');
  fclose(pf);
  cout<<"\r\nOke. Data sudah disimpan pada file.\r\n";

  getche();
}
```

SAMPLE PROGRAM :

```
main()
{
  FILE *pf;
  int nilai;
  int nomor = 0;
  clrscr();
  if((pf=fopen("BILANGAN.DAT","rb"))==NULL)
    {
     cout<<"  File gagal dibuka.\r\n";
        exit(1);
    }
  cout<<"\n Isi file BILANGAN.DAT : \r\n";
  while(1)
    {
    nilai = getw(pf);
    if(feof(pf) != NULL) break;
              cout<<" "<<++nomor<<" "<< nilai<<"\r\n";
     }
  fclose(pf);
  getche();
}
```

- **Reading File per Blok**

  To save or read data files in the form of unitary block (number of bytes) , for example, to type float or struct (structure)

```
int fread(void *buffer, int n, FILE *ptr_file);
```

- **Saving File per Blok**

```
int fwrite(void *buffer, int jum_byte, int n, FILE *ptr_file);
```

SAMPLE PROGRAM :

```
main()
{
  FILE *f_struktur;
  char jawab;

  struct data_pustaka
    { char judul[26];
            char pengarang[20];
            int jumlah;
    } buku;

  if((f_struktur=fopen("DAFBUKU.DAT","wb"))==NULL)
  {
    cout<<"File tak dapat diciptakan !!\r\n";
    exit(1);
  }
  do
  {
   clrscr();
   cout<<"  Judul buku        :  "; cin>>buku.judul;
   cout<<"  Nama pengarang    :  "; cin>>buku.pengarang;
   cout<<"  Jumlah buku       :  "; cin>>buku.jumlah;
   fflush(stdin);
   fwrite(&buku, sizeof(buku), 1, f_struktur);
struktur*/
   cout<<"\r\nMau merekam data lagi (Y/T) : "; jawab = getche();
  }
  while(jawab =='Y' || jawab =='y');
  fclose(f_struktur);
}
```

- **Reading Data String from File**

The function that is used to read data on file ie string fgets( ) to store the string str into a file . And fputs( ) to read a string from the file until ditemukannaya new - line character ' \ n ' or after the (n - 1) character, invitation n is the maximum length of the string that is read by a read.

```
char *fgets(char *str, int n, FILE *ptr_file);
```

- **Saving Data String dari File**

```
int fputs(char *str, FILE *ptr_file);
```

SAMPLE PROGRAM :


```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

#define PANJANG 256
main()
{
  FILE *f_teks;
  char string[PANJANG];
  char namafile[65];
  clrscr();
  cout<<"\n PROGRAM UNTUK MELIHAT ISI FILE TEKS\r\n";
  cout<<"\n Nama file : ";
  gets(namafile);

  if((f_teks=fopen(namafile, "rt"))==NULL)
  {       cout<<"File tak dapat dibuka\r\n";
   exit(1);
  }

while((fgets(string, PANJANG, f_teks)) != NULL)
  cout<<"\n Isi file :\n";
  cout<<"  "<<string<<"\r";
  fclose(f_teks);
  getche();
}
```

Renaming a File

A useful function to replace files, the rename( ).

Form of the declaration :

```
int rename(char *namafilelama, char *namafilebaru);
```

SAMPLE PROGRAM :

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
main()
{
  int kode;
  char namafilelama[65], namafilebaru[65];

  clrscr();

  cout<<"\n  Nama file yang akan diganti : ";
  cin>>namafilelama;
  cout<<"\n  Nama file pengganti        : ";
  cin>>namafilebaru;

kode = rename( namafilelama, namafilebaru );
  if( kode == -1 )
     cout<<"  Gagal dalam mengganti nama file !!\r\n";
  else
     cout<<"  Ok.Nama file sudah diganti !!\r\n";
  getche();
}
```

File (3)

Algorithm and Programming II

Wilis Kaswidjanti
Informatika UPN "Veteran" Yk

A. Consolidation algorithm

- Consolidation algorithm is a grouping of data with the same key that must be processed as a single unit.
- Example :
  - Given a value of Student Records, the student can have some fruit value (for a semester taking a few courses, and each student can be different subject).
  - Make algorithm to calculate the average value of each student , and create a list of values that is simpler, which write the NIM and the average value of each student.

FILE 1

NIM	Nilai
70001	50
70001	80
70001	60
70001	70
70002	55
70002	75
70002	65
70002	86
70002	64
70003	77
70003	83
70004	45
70004	54
70004	60

FILE 2

NIM	Nilai Rata-rata
70001	65
70002	69
70003	80
70004	53

Consolidation Algorithm

{Initial conditions : ArsipMhs already contains NIM and value }

{The final condition : record already grouped by NIM same , the value is the average value }

Deklarasi

```
Type DataMhs : Record
        < NIM : integer
          Nilai : real >
ArsipMhs1, ArsipMhs2 : seqFile of DataMhs
Mhs : DataMhs
CurrentNim, JumNil, nMK : integer
Rata : real
Function Mark(Input Mhs : DataMhs) → Boolean
{tanda akhir = <999,'xxxxx',99.9>}
```

Deskripsi

```
Open(ArsipMhs1,1)
    Open(ArsipMhs2,2)
    Fread(ArsipMhs1, Mhs)
    If (Mark(Mhs) = true) then   // if (Mhs.NIM ==9999) then
      Output('Arsip kosong...')
    Else
      While (Mark(Mhs) = false) do  // (Mhs.NIM <> 9999)
            JumNil ← 0
            nMK ← 1
            currentNIM ← Mhs.NIM {record1 dari ArsipMhs1}
            Repeat
                        JumNil ← JumNil + Mhs.Nilai
                        Fread(ArsipMhs1,Mhs)
                        nMK ← nMK + 1
            Until (currentNIM <> Mhs.NIM)
            Rata ← JumNil/nMK
              Mhs.NIM ← currentNIM
              Mhs.Nilai ← Rata
            Fwrite(ArsipMhs2, Mhs)
            Output(currentNIM,Rata)
      Endwhile
          Mhs.NIM ← 9999
          Mhs.Nilai ← 999
          Fwrite(Arsip2,Mhs)  {mark}
    EndIf
    Close(ArsipMhs1)
    Close(ArsipMhs2)
```

B. Processing Two File Streak

- **1. *Merging***
- Merging is merging two files that recordnya same type.
- To perform merging two ways . The simplest way is the second file data is added after the last record in the first file , thus forming a new file .
- This method can not be used if the key field both files already sorted, the desired file key field combination that is also ordered

Example :

NIM	Nama
70001	Adi
70003	Budi
70004	Susi
70006	Anti
70008	Doni

File ArsipMhs1

NIM	Nama
70002	Yudi
70009	Eka
70010	Bima

File ArsipMhs2

NIM	Nama
70001	Adi
70002	Yudi
70003	Budi
70004	Susi
70006	Anti
70008	Doni
70009	Eka
70010	Bima

File ArsipMhs3

- **MergingSambung Algorithm**

{Combining two successive file is ArsipMhs1 and ArsipMhs2, into a new file that is ArsipMhs3, by the way, all records connected to the second record after the last record in the first archive}

{Initial conditions : first and second archive already contains data}

{The final condition : third archive archive containing the results of the second connection }

- ***Deklarasi:***

Type DataMhs : Record

<NIM : Integer,

Nama: String>

ArsipMhsl, ArsipMhs2, ArsipMhs3 : SeqFile of DataMhs

Mhs : DataMhs

Function Mark(Input Mhs : DataMhs) → Boolean

Deskripsi :

```
Open(ArsipMhs1, 1)
Open(ArsipMhs2, 1)
Open(ArsipMhs3, 2)
FRead(ArsipMhs1, Mhs)
While (Mark(Mhs) = false) Do
    FWrite(ArsipMhs3, Mhs)
    FRead(ArsipMhs1, Mhs)
EndWhile
FRead(ArsipMhs2, Mhs)
While (Mark(Mhs) = false) Do
    FWrite(ArsipMhs3, Mhs)
    FRead(ArsipMhs2, Mhs)
EndWhile
Mhs.NIM ← 99999
Mhs.Nama ← xxxxx
FWrite(ArsipMhs3, Mhs)
Close(ArsipMhsl)
Close(ArsipMhs2)
Close(ArsipMhs3)
```

Merging2 Algorithm {versi And}

Deklarasi:

Type DataMhs : Record

<NIM : Integer,

Nama : String>

ArsipMhs1, ArsipMhs2, ArsipMhs3 : SeqFile of DataMhs

Mhs1, Mhs2 : DataMhs

Function Mark(Input Mhs : DataMhs) → Boolean

Deskripsi:

```
Open(ArsipMhs1, 1)
Open(ArsipMhs2, 1)
Open( ArsipMhs3 , 2)
FRead(ArsipMhsl, Mhs1)
FRead(ArsipMhs2, Mhs2)
While (Mark(Mhsl) = false) And (Mark(Mhs2) =false) Do
    If (Mhsl.NIM <= Mhs2.NIM) Then
        FWrite(ArsipMhs3, Mhsl)
        FRead(ArsipMhsl, Mhsl)
    Else
        FWrite(ArsipMhs3, Mhs2)
        FRead(ArsipMhs2, Mhs2)
    Endif
EndWhile
{Mark(Mhs 1) = true atau Mark(Mhs2) = true}
{salin record yg tersisa, pada ArsipMhs 1 atau ArsipMhs2}
{jikayg tersisa adalahArsipMhsl} .
While (Mark(Mhsl) = false) Do
    FWrite(ArsipMhs3, Mhs1)
    FRead(ArsipMhsl, Mhsl)
EndWhile
{jika yg tersisa adalah ArsipMhs2}
While (Mark(Mhs2) = false) Do
    FWrite(ArsipMhs3, Mhs2)
    FRead(ArsipMhs2, Mhs2)
EndWhile
FWrite(ArsipMhs3, <99999, 'xxxxx'>)
Close(ArsipMhsl)
Close(ArsipMhs2)
Close(ArsipMhs3)
```

Merging3 Algorithm {versi Or}

Deklarasi:

Type DataMhs : Record

<NIM : Integer,

Nama: String>

ArsipMhsl, ArsipMhs2, ArsipMhs3 : SeqFile of DataMhs

Mhs 1, Mhs2 : DataMhs

Function Mark(Input Mhs : DataMhs) → Boolean

Deskripsi:

```
Open(ArsipMhsl, 1)
Open(ArsipMhs2, 1)
Open(ArsipMhs3, 2)
FRead(ArsipMhs1, Mhs1)
FRead(ArsipMhs2, Mhs2)
While (Mark(Mhsl) = false) Or (Mark(Mhs2) = false) Do
    If (Mhsl.NIM <= Mhs2.NIM) Then
        FWrite(ArsipMhs3, Mhs1)
        FRead(ArsipMhsl, Mhsl)
    Else
        FWrite(ArsipMhs3, Mhs2)
        FRead(ArsipMhs2, Mhs2)
    Endif
EndWhile {Mark(Mhs 1) = true dan Mark(Mhs2) = true}
FWrite(ArsipMhs3, <99999, 'xxxxx'>)
Close(ArsipMhs1)
Close(ArsipMhs2)
Close(ArsipMhs3)
```

Updating

- Updating is the process of editing a record price (key field is not editable) in the master file with data from the transaction file.
- The following is a general algoitma to rejuvenate record in the master file (to be consecutive, field value keynya sequential rise but could not unique).
- This means that a record in the master file may experience one or more times rejuvenation.

Example :

- File rejuvenation savings balance at the Bank , with the agreement if a monetary value to a negative value update file , aqueous extraction, but if the value of money in precious Update File positive , meaning savings transaction

File Master

NoRek	Saldo
001	100000
003	150000
004	175000
006	180000
007	210000
008	112000
009	135000
010	210000
999	0

File Transaksi

NoRek	Saldo
001	+5000
001	-2000
003	+1000
004	-1500
004	+10000
004	-7500
006	+10500
006	-5000
007	-3000
999	0

File New Master

NoRek	Saldo
001	103000
003	151000
004	176000
006	185500
007	207000
008	112000
009	135000
010	210000
999	0

- **Updating** *{title algorithm}*

  {Changing one field contents of the master file based on data from the transaction file and then save your edits to the new master file}

  {Initial conditions : a variable already valued the contents of a key field in that record position will be changed, the key field already ordered ascend}

  {Kondisiakhir : new master file already contains data from the master file editing based file transaksi1 }

Deklarasi

Type DataSaldo : Record
< NoRek : Integer,
  Saldo : Longint >
Master, Transaksi, NewMaster : SeqFile of DataSaldo
Nasabah1,Nasabah2 : DataSaldo
NewSaldo : Longint
FunctionMark(InputNasabah : DataSaldo) →
  Boolean

Deskripsi:

```
Open(Master, 1)
Open(Transaksi, 1)
Open(NewMaster, 2)
FRead(Master, Nasabah1)
FRead(Transaksi, Nasabah2)
While (Mark(Nasabahl) = false) Do
While (Nasabah2.NoRek < Nasabahl.NoRek)
and (Mark(Nasabah2) = false) Do
    FRead(Transaksi, Nasabah2) {skip record dari file trans}
EndWhile
If (Nasabah2.NoRek = Nasabahl.NoRek) Then
      NewSaldo ← Nasabah1.Saldo {yg akan diedit}
      While (Nasabah2.NoRek = Nasabahl.NoRek)
      and (Mark(Nasabah2) = false) Do
         NewSaldo← NewSaldo+ Nasabah2.Saldo
         FRead(Transaksi, Nasabah2)
      EndWhile
      Nasabah1.Saldo ← NewSaldo
      FWrite(NewMaster, Nasabah1)
      Else
      FWrite(NewMaster, Nasabahl) .
Endif
FRead(Master, Nasabah1)
EndWhile
FWrite(NewMaster, <999, 0>)
Close(Master) Close(Transaksi)
Close(NewMaster)
```

Splitting

- Splitting is splitting a file into two or more new files . The algorithm depends on the criteria of the solution.

Example :

Splitting the value of a course file a class based on the value of > = 55 and that the value of < 55.

NIM	Nilai
-	80
-	42
-	60
-	56
-	55
-	71
	38
	65
	40
	54
99999	999

NIM	Nilai
-	80
-	60
-	56
-	55
-	71
-	65
99999	999

NIM	Nilai
-	42
-	38
-	40
-	54
99999	999