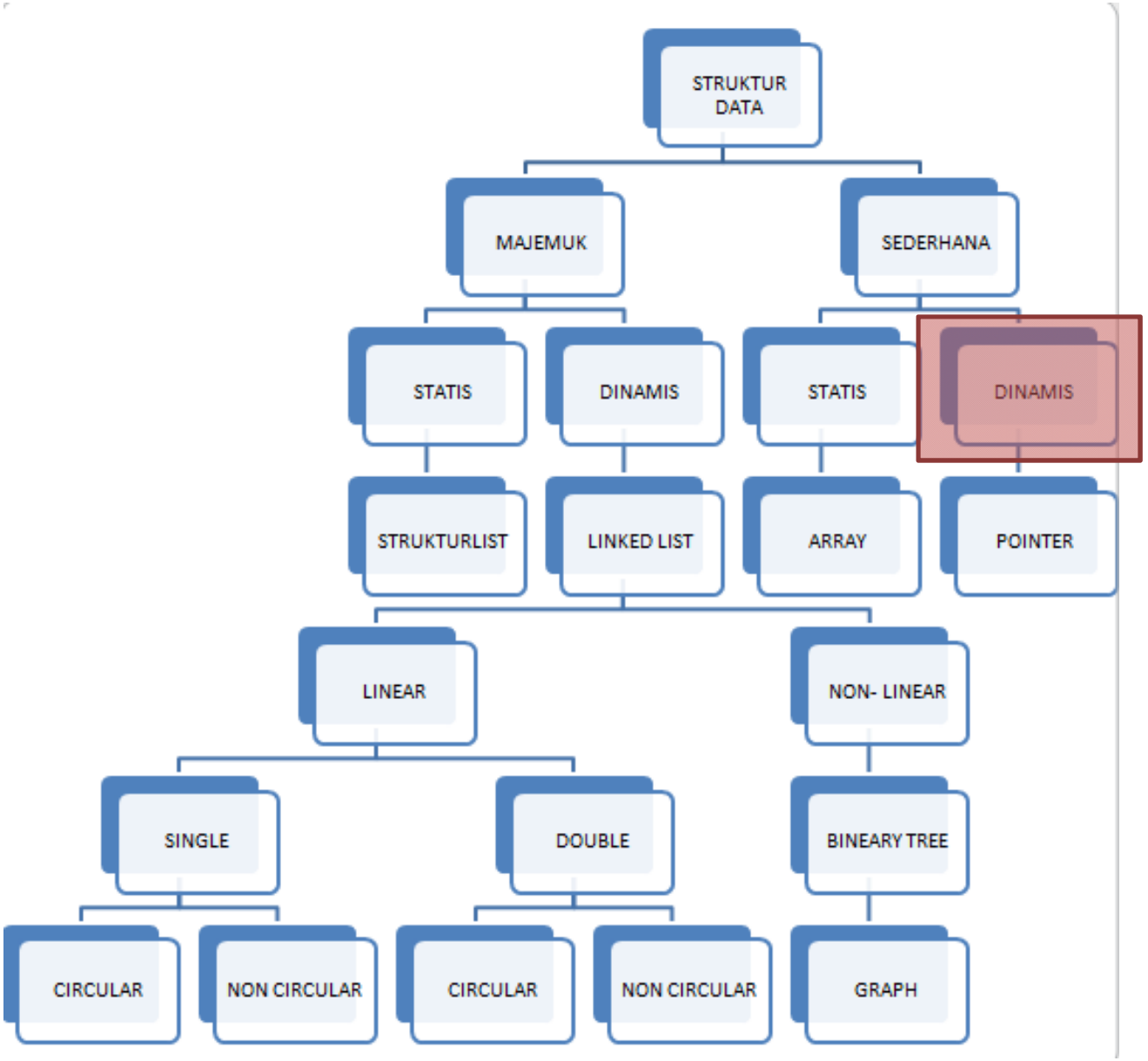
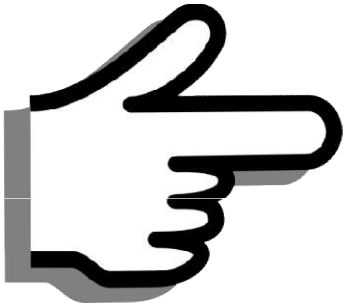
pointers

Agus Priyanto
www.ittelkom-pwtac.id


STRUKTUR DATA
MAJEMUK
SEDERHANA
STATIS
DINAMIS
STATIS
DINAMIS
STRUKTURLIST
LINKED LIST
ARRAY
POINTER
LINEAR
NON- LINEAR
SINGLE
DOUBLE
BINEARY TREE
CIRCULAR
NON CIRCULAR
CIRCULAR
NON CIRCULAR
GRAPH



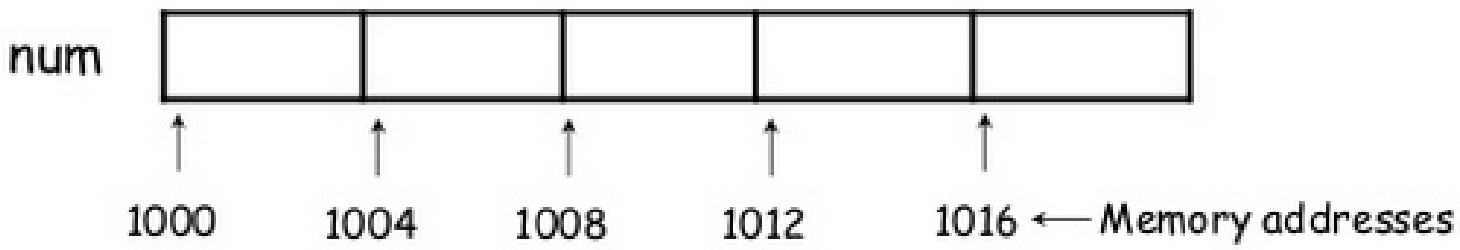
Mengapa pointer disebut dinamis?

Array

-Sebuah urutan variabel dengan nama dan tipe data yang sama

int num [5]

-5 buah urutan variabel dengan tipe integer dengan nama variabel num



- bersifat statis (ukuran dan urutannya sudah pasti).
- ruang memori yang dipakai olehnya tidak dapat dihapus bila variabel bertipe array tersebut sudah tidak digunakan lagi pada saat program dijalankan.
- **pointer** bersifat dinamis, variabel akan dialokasikan hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan dapat dialokasikan kembali.

Operator pointer

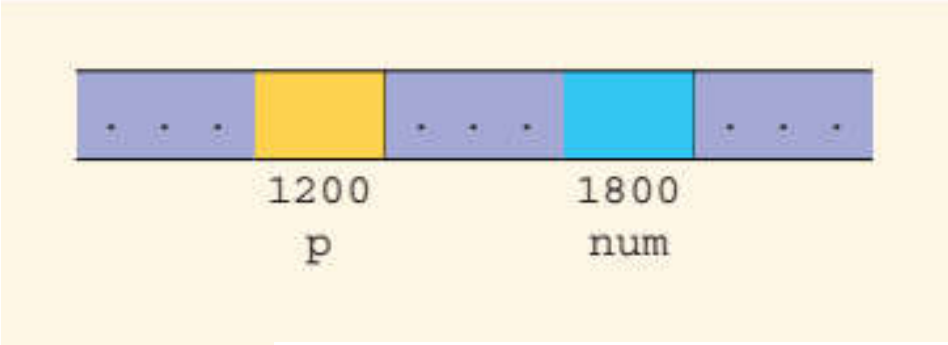
- & = menghasilkan **alamat**
- * = menghasilkan reference dari sebuah alamat (**nilai/value**)

Statement

- `int *p;`
  is equivalent to the statement:
  `int* p;`
  which is equivalent to the statement:
  `int * p`

- `int *p, q;`
  Of course, the statement:
  `int *p, *q;`

```
int *p;
int num;
```




```
1. num = 78;
2. p = &num;
3. *p = 2
```

After Statement	Values of the Variables	Explanation
1	. . . 78 . . . 1200 p, 1800 num	The statement **num = 78;** stores **78** into **num**.
2	. . . 1800 . . . 78 . . . 1200 p, 1800 num	The statement **p = #** stores the address of **num**, which is **1800**, into **p**.
3	. . . 1800 . . . 24 . . . 1200 p, 1800 num	The statement ***p = 24;** stores **24** into the memory location to which **p** points. Because the value of **p** is **1800**, statement 3 stores **24** into memory location **1800**. Note that the value of **num** is also changed.

```
#include <iostream>
using namespace std;
int main()
{
    int a[5];
    int *p;
    a[0]=24;
    a[1]=32;
    a[2]=81;
    a[3]=44;
    a[4]=23;
    p=&a[0];
    cout<<"alamat p : "<<p<<endl;
    cout<<"nilai p : "<<*p<<endl;
    return 0;
}
```

"E:\IT Telkom Purwokerto\Materi Perkuliahan\001-S1 Teknik Informatika\Algoritma

```
alamat p : 0x6afee8
nilai p : 24

Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```

Pointer Bertipe Void

- Pada C++ terdapat pointer yang dapat menunjuk ke tipe data apapun, pointer semacam ini dideklarasikan dengan tipe **void** sehingga sering dikenal dengan istilah **Void Pointer**.

```
#include <iostream>

using namespace std;

int main()
{
   void *p;
   int a=10;
   double b=23.4;
   char c='s';
   p=&a; //p menunjuk ke tipe data int
   cout<<"alamat (a=10) = "<<p<<endl;
   p=&b; //p menunjuk ke tipe data double
   cout<<"alamat (b=23.4) = "<<p<<endl;
   p=&c; //p menunjuk ke tipe data char
   cout<<"alamat (c='s') = "<<p<<endl;

   return 0
}
```

"E:\IT Telkom Purwokerto\Materi Perkuliahan\001-S1 Teknik Informatika\Algoritma dan Struktur Data\2018.2\Latihan\pointer

```
alamat (a=10) = 0x6afef8
alamat (b=23.4) = 0x6afef0
alamat (c='s') = 0x6afeef

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

Dynamic Variables

- **Operator new**
  The operator new has two forms: one to allocate a single variable and another to allocate an array of variables. The syntax to use the operator new is:

```
new dataType;              //to allocate a single variable
new dataType[intExp];      //to allocate an array of variables
```

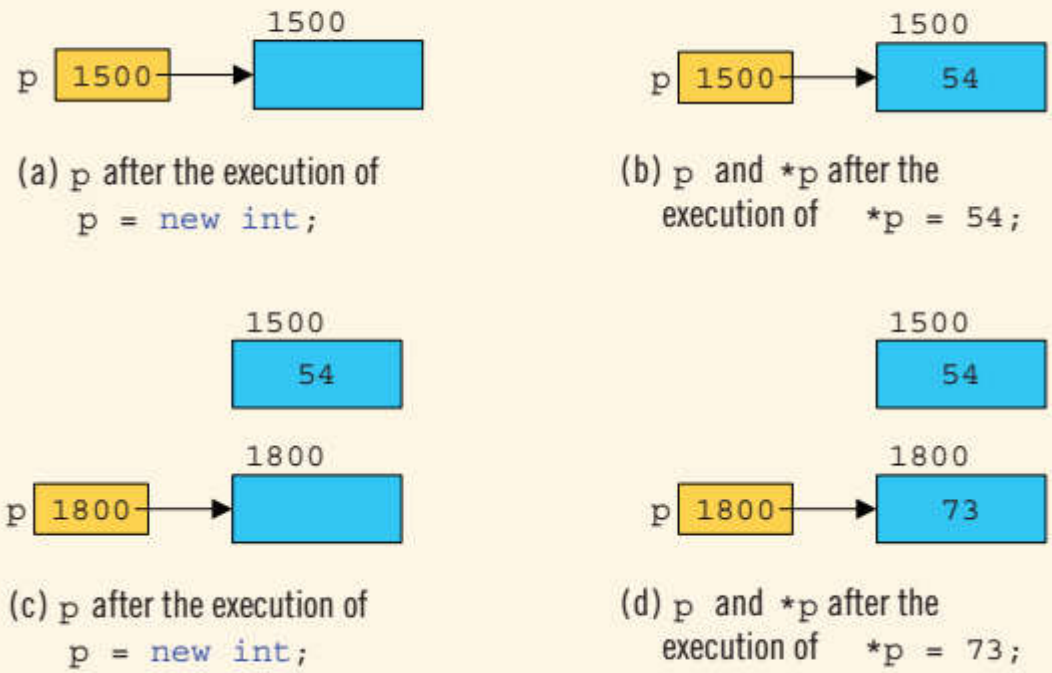
```
int *p;             //p is a pointer of type int
char *name;         //name is a pointer of type char
string *str;        //str is a pointer of type string

p = new int;        //allocates memory of type int
                    //and stores the address of the
                    //allocated memory in p
*p = 28;            //stores 28 in the allocated memory

name = new char[5];     //allocates memory for an array of
                        //five components of type char and
                        //stores the base address of the array
                        //in name
strcpy(name, "John");   //stores John in name

str = new string; //allocates memory of type string
                  //and stores the address of the
                  //allocated memory in str
*str = "Sunny Day";     //stores the string "Sunny Day" in
                        //the memory pointed to by str
```

- **Operator delete**

```
int *p;
```

This statement declares p to be a pointer variable of type int. Next, consider the following statements:

```
p = new int; //Line 1
*p = 54; //Line 2
p = new int; //Line 3
*p = 73; //Line 4
```



(a) p after the execution of
p = new int;

(b) p and *p after the
execution of *p = 54;

(c) p after the execution of
p = new int;

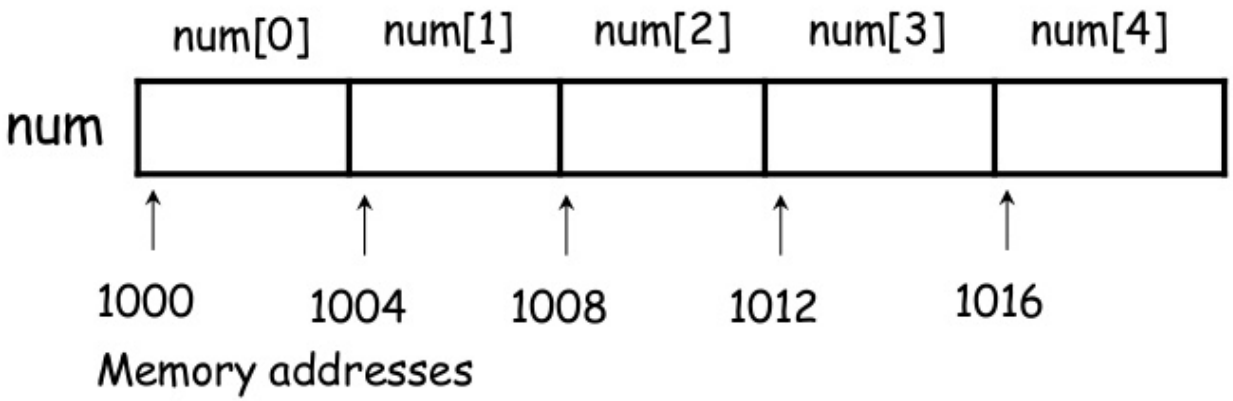
(d) p and *p after the
execution of *p = 73;

Hubungan array dan pointer

Manakah alamat masing-masing elemen?

```
int num[5] ;
```

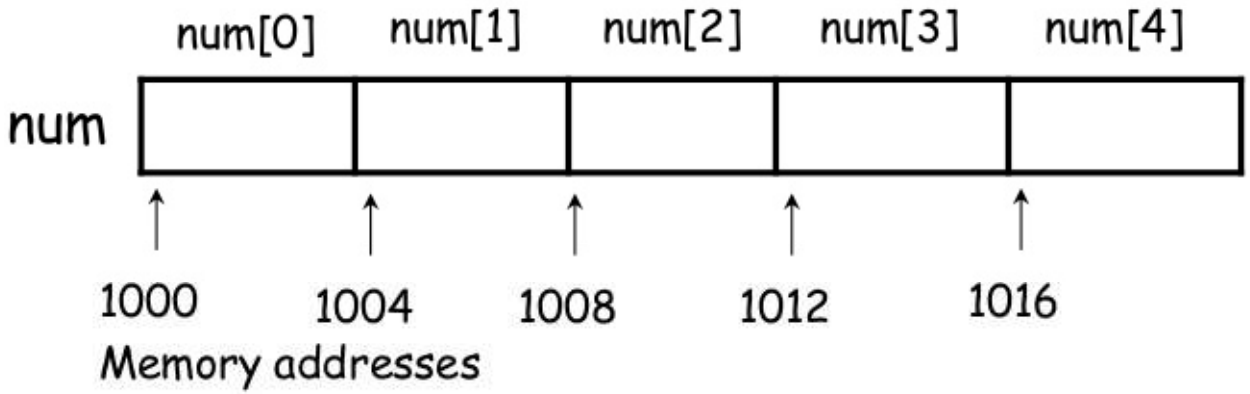
```
&num[0] == 1000
&num[1] == 1004
&num[2] == 1008
&num[3] == 1012
&num[4] == 1016
```



Hubungan array dan pointer

Apa itu num?

```
int num[5] ;
```



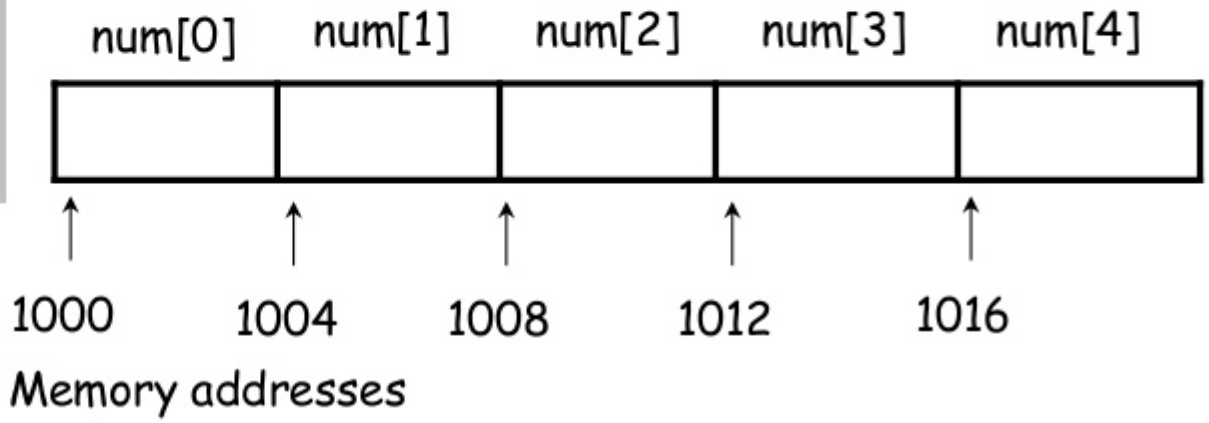
- *num* is the constant pointer of which value is the start address of the array.

```
num == &num[0] == 1000
```

- **Example : Arithmetic of pointers**
  - "*pointer* + 1" does not mean increasing *pointer* by 1.
  - "*pointer* + 1" is "the address of the next element".
  - "*pointer* – 1" is "the address of the prior element".

```
num == &num[0] == 1000

(num+0) == &num[0]
(num+1) == &num[1]
(num+2) == &num[2]
(num+3) == &num[3]
(num+4) == &num[4]
```

```
int num[5] ;
```



Pointer arithmetic

- **Example : Arithmetic of pointers**

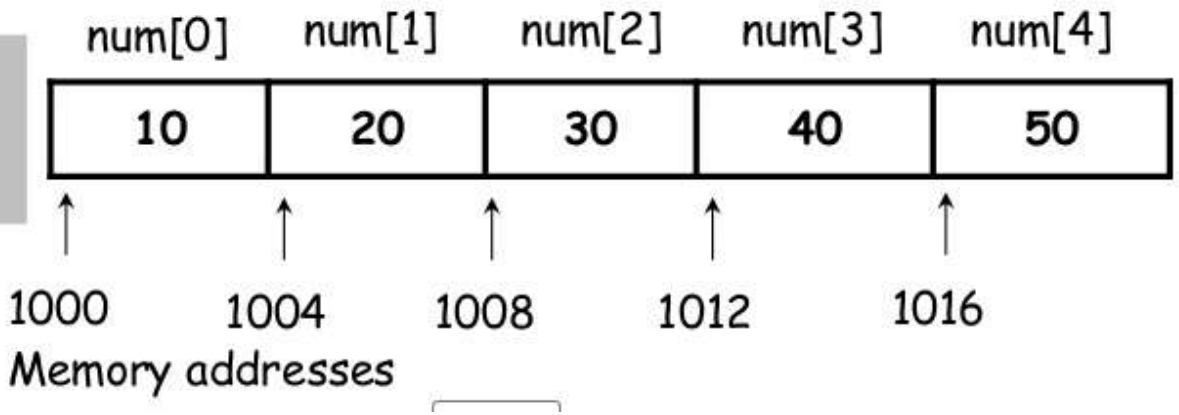
```
int num[5] ;

num[0] =10 ;
num[1] = 20 ;
num[2] = 30 ;
num[3] = 40 ;
num[4] = 50 ;
```

```
int num[5] ;

*num = 10 ;
*(num+1) = 20 ;
*(num+2) = 30 ;
*(num+3) = 40 ;
*(num+4) = 50 ;
```

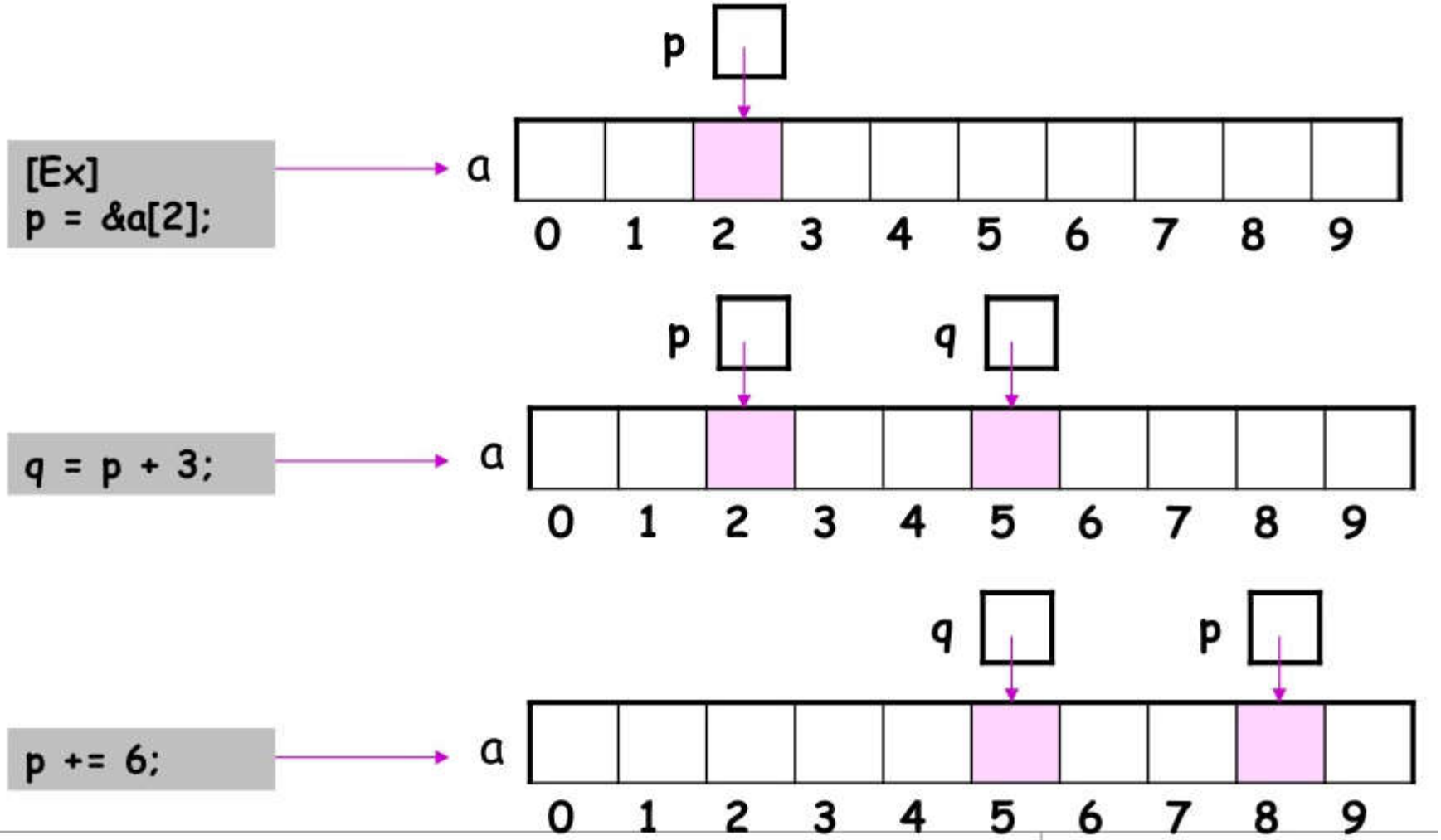
```
int num[5], *p = num ;

*p = 10 ;
*(p+1) = 20 ;
*(p+2) = 30 ;
*(p+3) = 40 ;
*(p+4) = 50 ;
```

```
int num[5], *p = num ;

p[0] = 10 ;
p[1] = 20 ;
p[2] = 30 ;
p[3] = 40 ;
p[4] = 50 ;
```

```
int num[5] ;
```

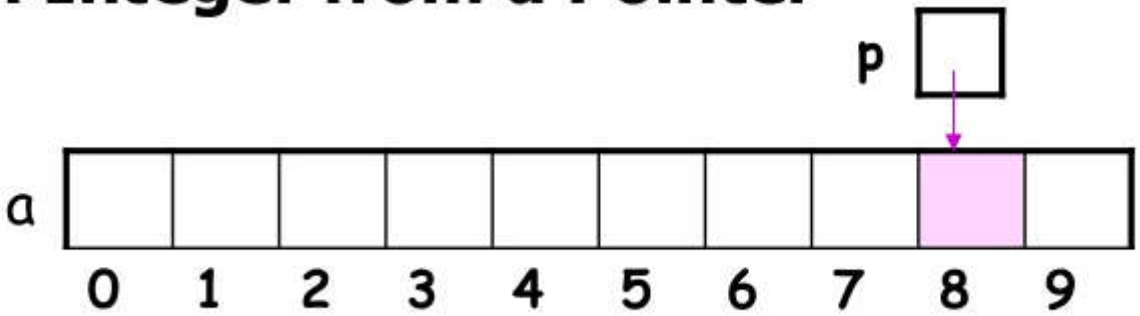


- **Adding an Integer to a Pointer**

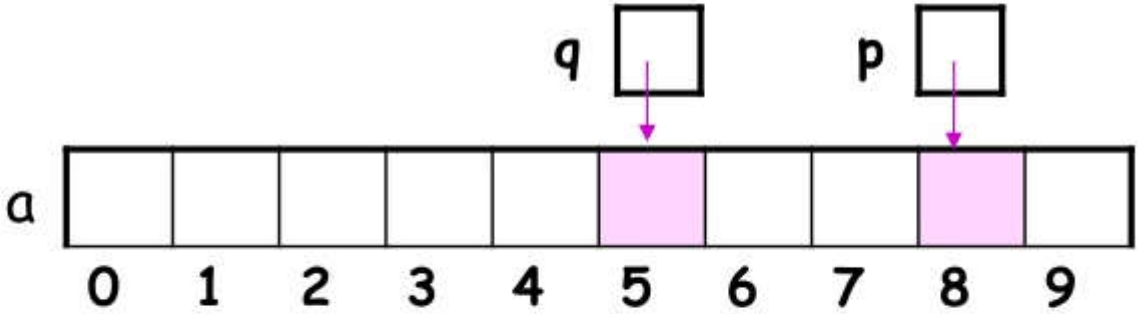


- **Subtracting an Integer from a Pointer**

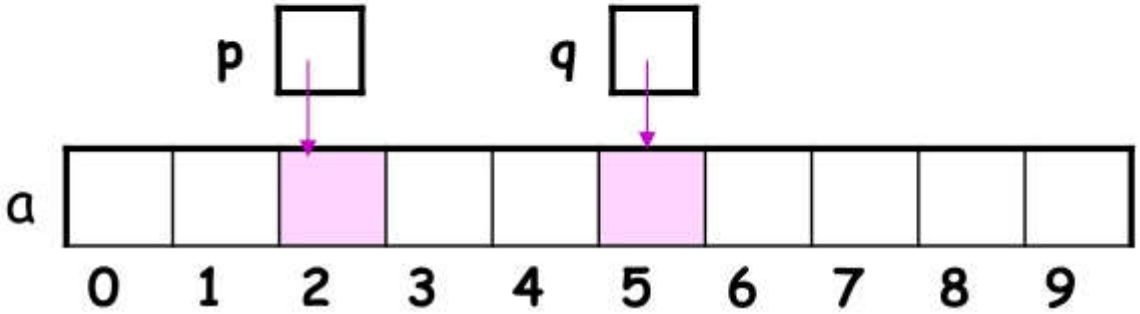
```
[Ex]
p = &a[8];
```




```
q = p - 3;
```

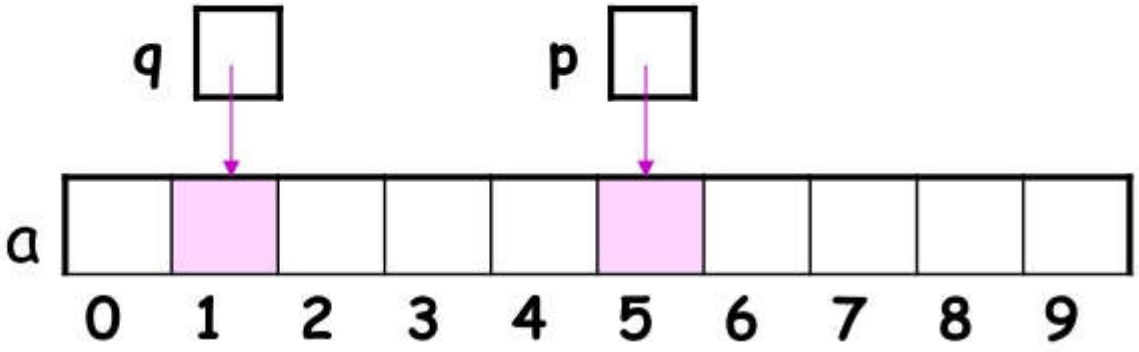



```
p -= 6;
```



- **Subtracting Pointers**

```
[Ex]
p = &a[5];
q = &a[1];

i = p - q;          /* i == 4  */
i = q - p;          /* i == -4 */
```



- **Comparing Pointers**
  - Relational operators (<, <=, >, >=) can be applied
  - Equality operators (==, !=) can be applied

```
[Ex]
p = &a[5];
q = &a[1];

p <= q;    /* result is 0 */
p >= q;    /* result is 1 */
```

- **Example: Pointer Operation**

```
int a[ ] = { 5,15,25,43,12,1,7,89,32,11}
int *p = &a[1], *q = &a[5] ;

1. *(p + 3) ?

2. *(q - 2) ?

3. q - p ?

4. if ( p > q ) ?

5. if ( *p > *q )?
```

bila menggunakan **pointer** dengan cara yang salah maka akan menyebabkan **sistem operasi menjadi rusak.** **Jadi, berhati-hatilah…..**

Thank you!