EEM103
Computer Programming

Week10

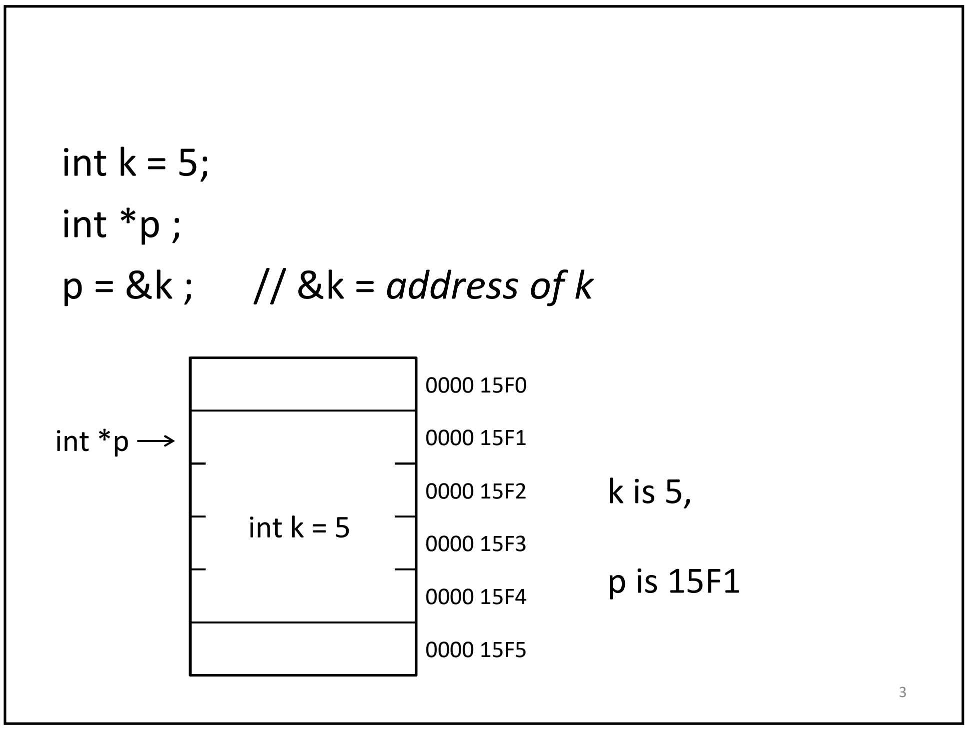
- Pointers
- Pointer operators
- Pointer arithmetic
- Array-Pointer relation

Pointers

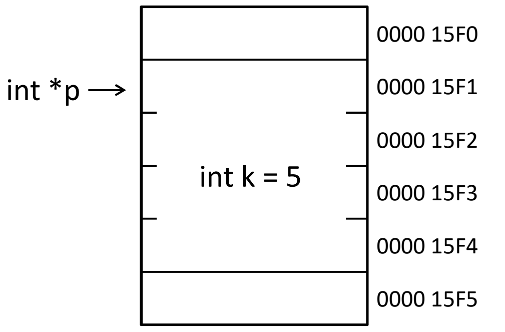
- **Pointers** are variables whose values are memory addresses.
  - A variable directly contains a specific value.
  - A pointer contains an address of a variable that contains a specific value.
- Declaring Pointers;

  type * pointer_name;
- e.g;

```
int * p;        /* p is a pointer which keeps address of an integer.*/
float * q;      /* q is a pointer which keeps address of a float.*/
int * p2, i;    /* p2 is an int pointer, i is an int variable*/
```



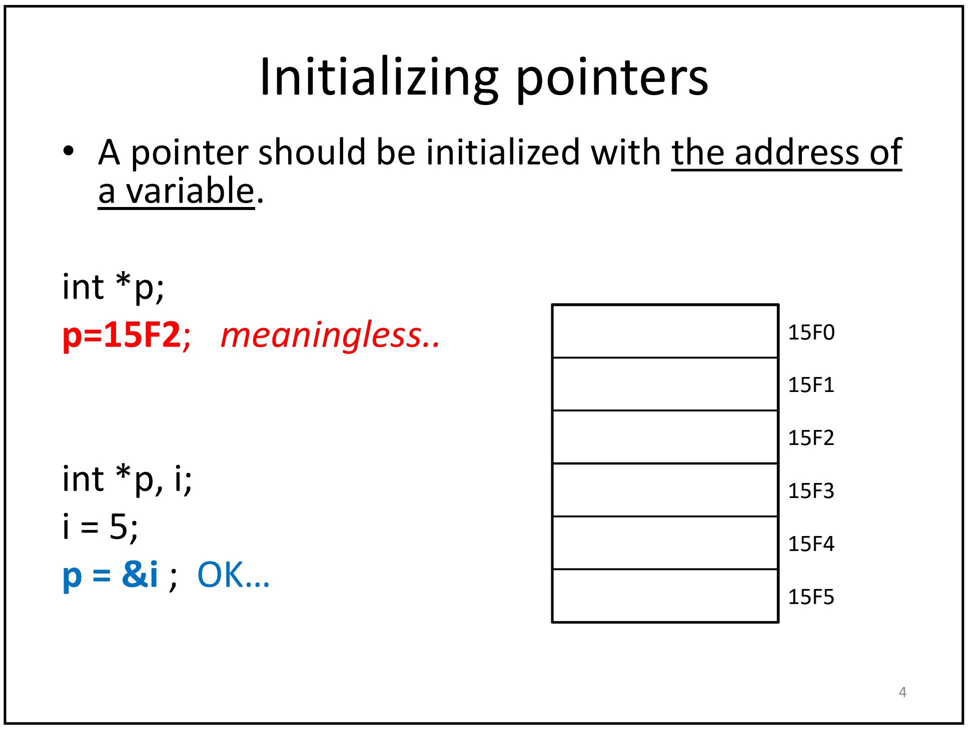
```
int k = 5;
int *p ;
p = &k ;      // &k = address of k
```



k is 5,

p is 15F1

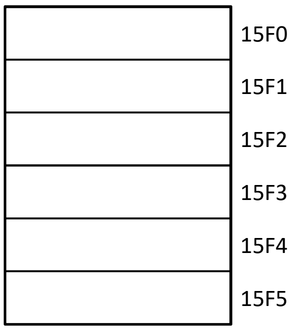
3



Initializing pointers

- A pointer should be initialized with the address of a variable.

int *p;
p=15F2; *meaningless..*



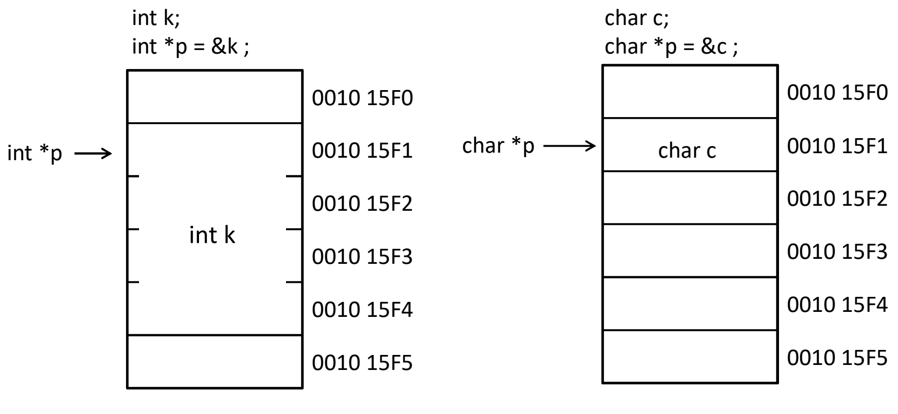
int *p, i;
i = 5;
p = &i ; OK…

4

Size of a pointer

- All pointers have the same size,
  - which is equla to the addresing type of the operating system, (32 or 64 bit)
- But, the type of the pointer determines the size of memory area that it effects.

5



printf("%p",p); => 0010 15F1

printf("%p",p); => 0010 15F1

Pointer operators

1. address-of operator: **&**
2. indirection (dereferencing) operator: *****

int *p ; → int pointer p

*p → content of p (the value of the integer that p points)

int i; → i is an integer variable

&i → address of i

7

int i;
int *p;

p=&i;
i=5;   *p=5;   → *same*

```
printf(“%d\n”, p);
printf(“%d\n”, *p);
printf(“%d\n”, i);
printf(“%d\n”, &i);
```

8

Pointer Arithmetic

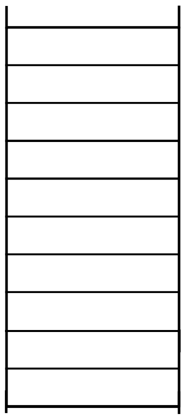
int i, *p;

i=5;

p=&i; → assume, p=1F51

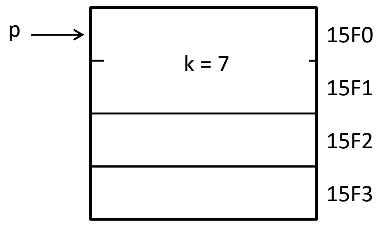
p=p+1; → p = 1F51 + **4** = 1F55

Why !!

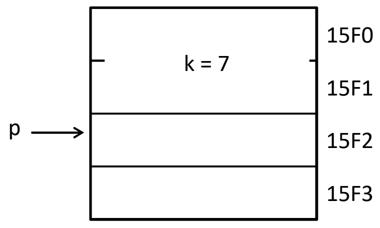


short int k = 7;

short int * p = &k;



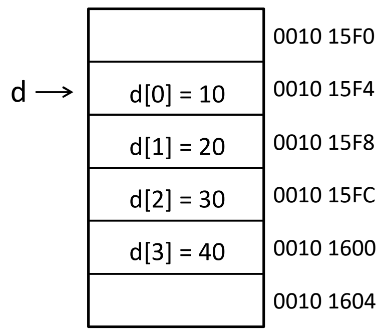
p++ ;



Array – Pointer relation

- **<u>Name of an array</u> is a <u>POINTER</u>.**
  - and, it points the first element of the array.

e.g: int d[4] = {10,20,30,40} ;



```
printf("%d \n" ,  d[0]  ) ;
printf("%d \n" ,  *d  ) ;

printf("%p \n" ,  &d[0] ) ;
printf("%p \n" ,  d ) ;

printf("%d \n" ,  *(d+2)  ) ;
printf("%d \n" ,  d[2] ) ;
printf("%p \n" ,  d+2  ) ;
```

11