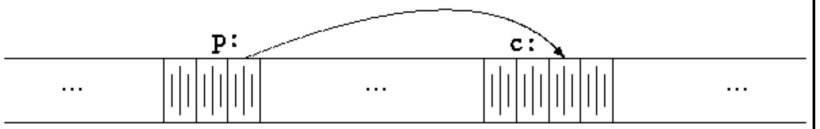
CSE 101
Slide Set 12

Doç. Dr. Mehmet Göktürk
Department of Computer Engineering

1

Pointers

- A typical machine has an array of consecutively numbered or addressed memory cells that may be manipulated individually or in contiguous groups.
- One common situation is that any byte can be a char, a pair of one-byte cells can be treated as a short integer, and four adjacent bytes form a long.
- A pointer is a group of cells (often two or four) that can hold an address. So if c is a char and p is a pointer that points to it, we could represent the situation this way:




```
p = &c;
```


2

Pointers

p = &c;

assigns the address of c to the variable p,
and p is said to ``point to'' c.

The & operator only applies to objects in memory: variables and array elements.

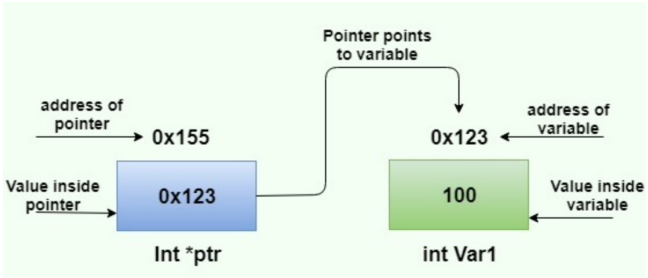
3

*

- The unary operator * is the indirection or dereferencing operator;
- when applied to a pointer, it accesses the object the pointer points to.

4

Pointers



5

& and * example

- Suppose that x and y are integers and ip is a pointer to int.

```
int x = 1, y = 2, z[10];
int *ip;          /* ip is a pointer to int */

ip = &x;          /* ip now points to x */
y = *ip;          /* y is now 1 */
*ip = 0;          /* x is now 0 */
ip = &z[0];       /* ip now points to z[0] */
```


6

The use of pointers

f ip points to the integer x, then *ip can occur in any context where x could, so

```
*ip = *ip + 10;
```

increments *ip by 10.
The unary operators * and & bind more tightly than arithmetic operators, so the assignment

```
y = *ip + 1
```

takes whatever ip points at, adds 1, and assigns the result to y, while

```
*ip += 1
```

increments what ip points to, as do

```
++*ip
```

and

```
(*ip)++
```

The parentheses are necessary in this last example; without them, the expression would increment ip instead of what it points to, because unary operators like * and ++ associate right to left.

7

The use of pointers

since pointers are variables, they can be used without dereferencing.

For example, if iq is another pointer to int,

```
iq = ip
```

copies the contents of ip into iq, thus making iq point to whatever ip pointed to.

8

Pointers and Function Arguments

- Since C passes arguments to functions by value, there is no direct way for the called function to alter a variable in the calling function.
- For instance, a sorting routine might exchange two out-of-order arguments with a function called swap. It is not enough to write :

```
swap(a, b);
```


9

Pointers and Function Arguments

- Since C passes arguments to functions by value, there is no direct way for the called function to alter a variable in the calling function.
- For instance, a sorting routine might exchange two out-of-order arguments with a function called swap. It is not enough to write :

```
void swap(int x, int y)   /* WRONG */
{
    int temp;

    temp = x;
    x = y;
    y = temp;
}
```

Because of call by value, `swap` can't affect the arguments `a` and `b` in the routine that called it. The function above swaps copies of `a` and `b`.

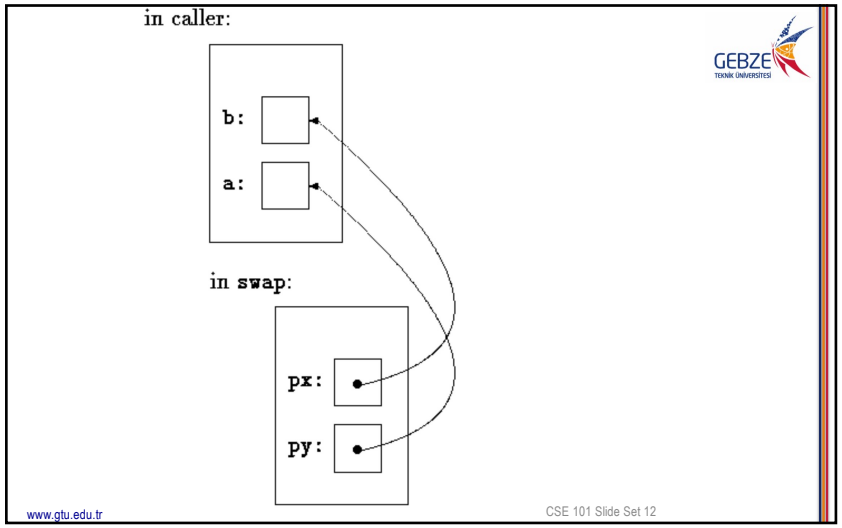
10

Correct Way !

- The way to obtain the desired effect is for the calling program to pass pointers to the values to be changed

```
swap(&a, &b);
```


11



12

Correct Way !

```
void swap(int *px, int *py)  /* interchange *px and *py */
{
    int temp;

    temp = *px;
    *px = *py;
    *py = temp;
}
```


13

Examples

14

15

16

Pointers and Arrays

```
int a[10];
```

17

Pointers and Arrays

```
int a[10];
```

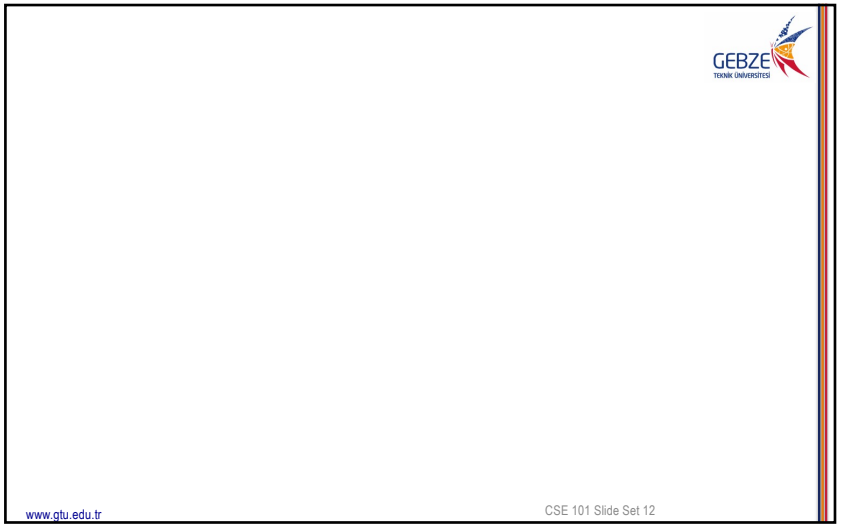

```
int *pa;
pa = &a[0];
```

pa[i] is identical to ***(pa+i)**

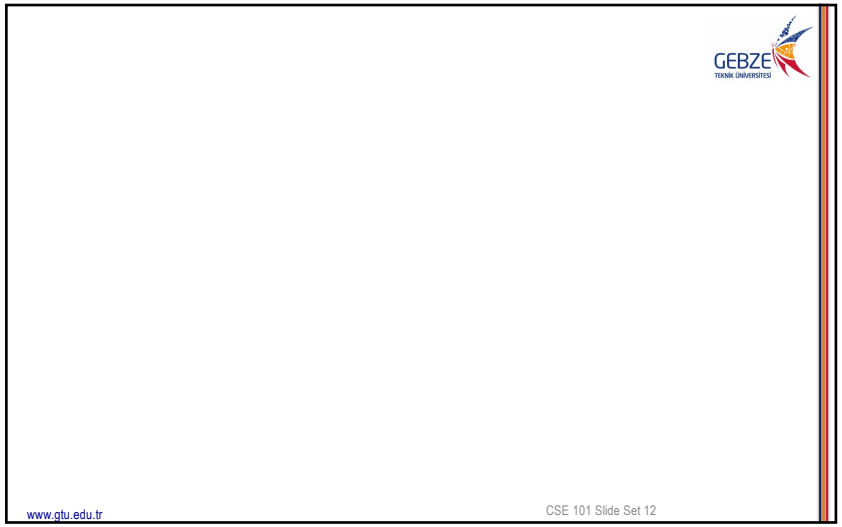
18

19

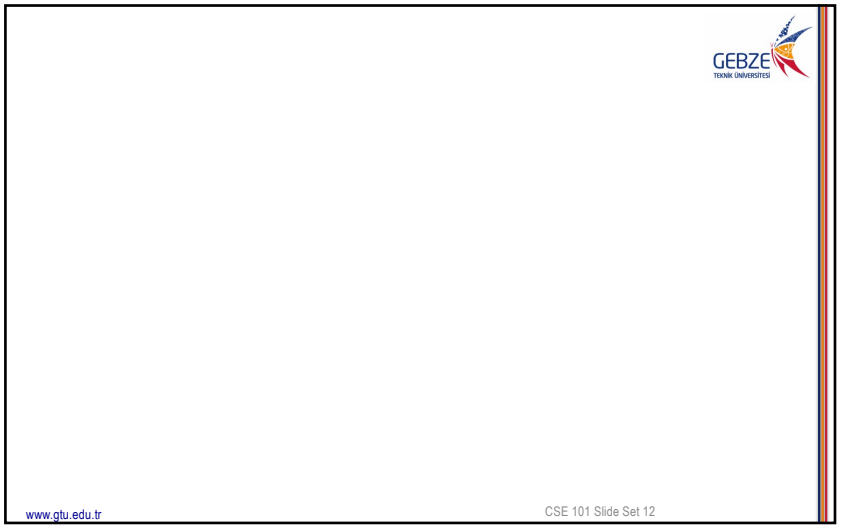
20


GEBZE
TEKNİK ÜNİVERSİTESİ
www.gtu.edu.tr
CSE 101 Slide Set 12

21


GEBZE
TEKNİK ÜNİVERSİTESİ
www.gtu.edu.tr
CSE 101 Slide Set 12

22


GEBZE
TEKNİK ÜNİVERSİTESİ
www.gtu.edu.tr
CSE 101 Slide Set 12

23