- Now it is time to start POINTERS!!!

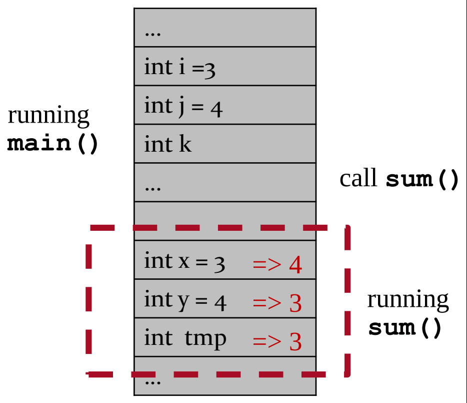
Pointers K&R Ch 5

- Basics: Declaration and assignment (5.1)
- Pointer to Pointer (5.6)
- Pointer and functions (5.2)
- Pointer arithmetic (5.4)
- Pointers and arrays (5.3)
- Arrays of pointers (5.6)
- Command line argument (5.10)
- Pointer to arrays and two dimensional arrays (5.9)
- Pointer to functions (5.11)
- Pointer to structures (6.4)
- Memory allocation (extra)

Calling-by-Value

- In C, all functions are **called-by-value**
  - Value of the arguments are passed to functions, but not the arguments themselves (call by reference)

```
int swap (int x, int y)
{ int tmp;
  tmp = x;
  x = y;
  y = tmp;
}
main(){
  int i=3, j=4;
  swap(i,j)
}
```

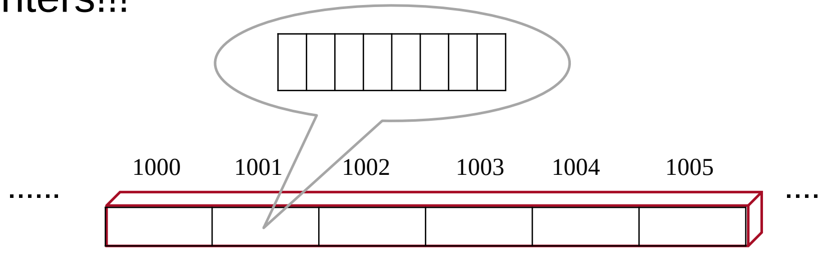


```
char [] fromStr = "Hello!"
char [20] toStr;
strcpy(toStr, fromStr);

fgets(toStr, 10, stdin);
```

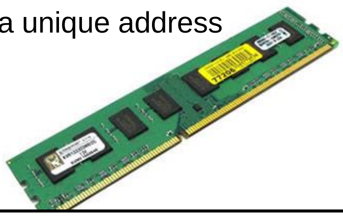
- Given an array as an argument, a function can modify the contents of the array -- Arrays are passed *as if* “call-by-reference”
  - Also `scanf ("%d %s", &a, arr);`

- But isn’t C “call-by-value”? -- pass single numerical value
  - How to pass the strings to `strcpy()`
  - How does `strcpy(),strcat(), scanf(),fgets()` etc modify argument?

Pointers!!!



- computers memory
  - Thousands of sequential storage location byte (8 bits)
  - Range 0 – max
  - Each byte has a unique address



5

5

* & specifically for pointers

```
int  * p ;
```

- p is a pointer variable capable of pointing to variable of type int – storing the address of a int variable

```
int * p, *q;
int  j, a[10],  * p, *q;
```

```
&x
```

- address of a variable, array element. (No expression)

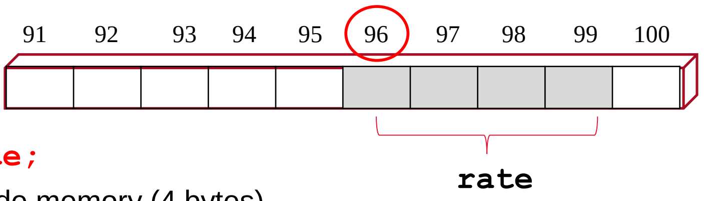
```
p = &x
int *p2 = &x;     scanf("%d %d",  &a, &b);

p = &arr[0]; // later
```

6

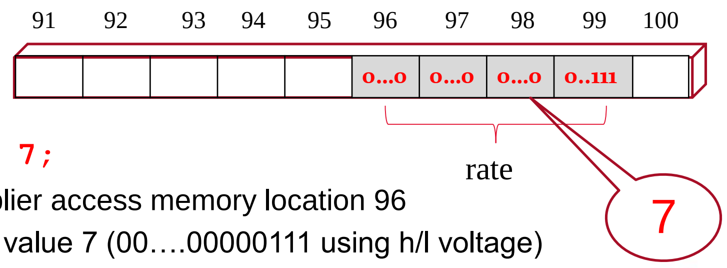
6

```
#include <stdio.h>



main()
{
 int a, b;
 printf("pls enter two numbers separated by blank: " );

 scanf( "%d  %d",  &a, &b);    /* assign value to a b  */

 printf("input %d and %d. sum is %d\n", a, b, (a+b)  );
}
```

7



- **int rate;**
  - set aside memory (4 bytes)
  - associates 96 (starting address) with **rate**;



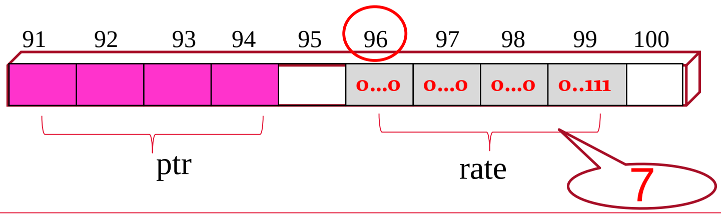
- **rate = 7;**
  - Complier access memory location 96
  - Store value 7 (00….00000111 using h/l voltage)
  - Hidden from you

8

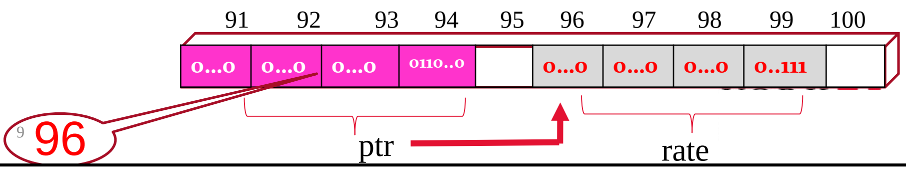
8

Declare and initialize pointer

- `int *ptr;   /* declare a pointer to int  */`
  - Create a variable holding the address of other variable



- `ptr = &rate   /*assigning address of rate*/`
  - Store address/pointer of rate in ptr (ptr’s value is the address)
  - ptr now ‘points to’ rate

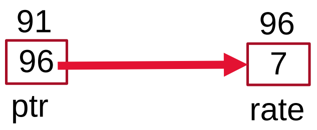


9

```
int *ptr;        /* I’m a pointer to an int */
```

91
ptr
96
7
rate

```
ptr= &rate;  /*I got the address of rate */
```



```
*ptr;      /* dereferencing. Indirect access.
              Get contents of the pointee*/
```

ptr   &rate --- address of rate

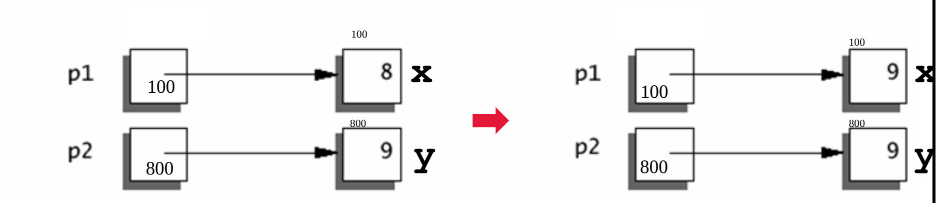
* ptr   rate --- content (value) of rate   “mnemonic”

```
printf("%d", rate);    // 7  "direct access"
printf("%d", *ptr);   //  7  "indirect access"
```

10

Some example of Pointers

```
int *p1, *p2;  int x = 8, y = 9;
p1 = &x;  p2 = &y;
*p1 = *p2;    // x = y
```



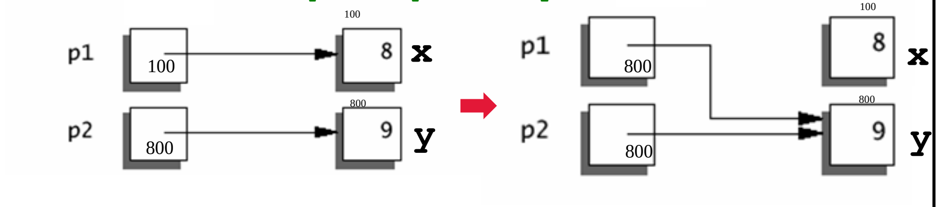
```
// copy value of p2's pointee (y) into pointee of p1 (x)
  *p1 is the alias of x    *p2 is the alias of y
```

11

11

Some example of Pointers

```
int *p1, *p2;  int x = 8, y = 9;
p1 = &x;  p2 = &y;
p1 = p2;   /*copy the content of p2 (address of y) into p1
            now p1 also points to y  */
```



Java:
```
Student s1 = new Student("John", 22);
Student s2 = s1;
```

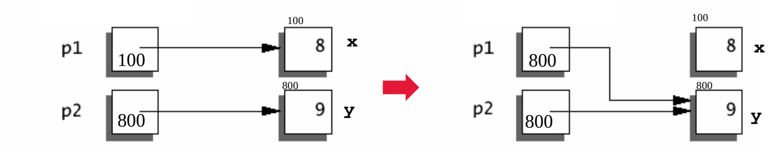
12

12

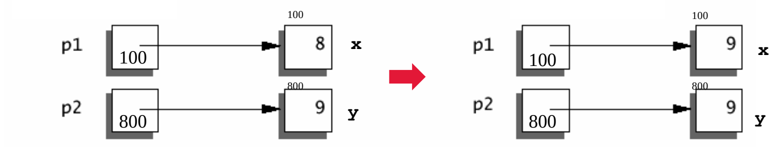
Some example of Pointers -- summary

```
int *p1, *p2, x = 8, y = 9;
p1 = &x;  p2 = &y;
```

```
p1 = p2;    // p1 = &y
```



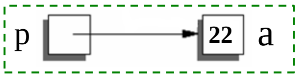
```
*p1 = *p2;    // x = y
```

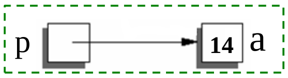


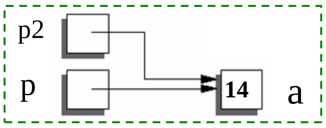
13

13

```
int main()
{
  int a = 22;
  int *p = &a;
  printf("%d %d\n", a, *p);  /* 22 22 */

  *p =  14;  // a = 14
  printf("%d %d\n", a, *p);  /* 14 14 */

  int *p2 = p;


  (*p2)--;  // *p2 = *p2 - 1;
  printf("%d %d %d\n", a, *p, *p2);
}                      /* 13   13  13  alias */
```







14

14