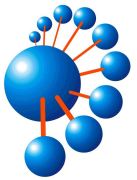
Programación I
2021-2

Clase 9
Punteros


Universidad de Concepción

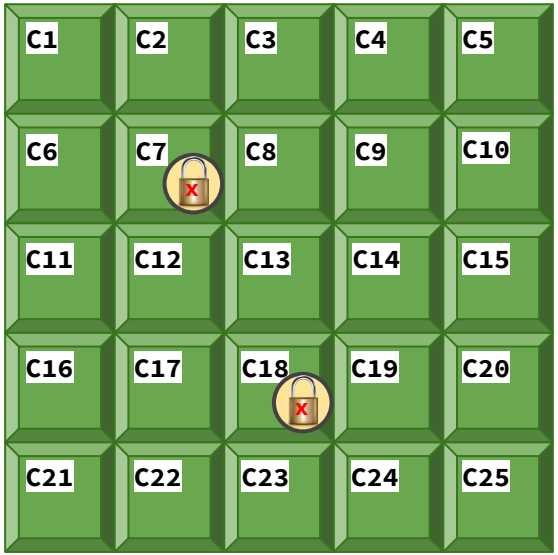
José Fuentes - jfuentess@inf.udec.cl

Departamento de Ingeniería Informática y Ciencias de la Computación


Punteros en C: Una analogía con casilleros

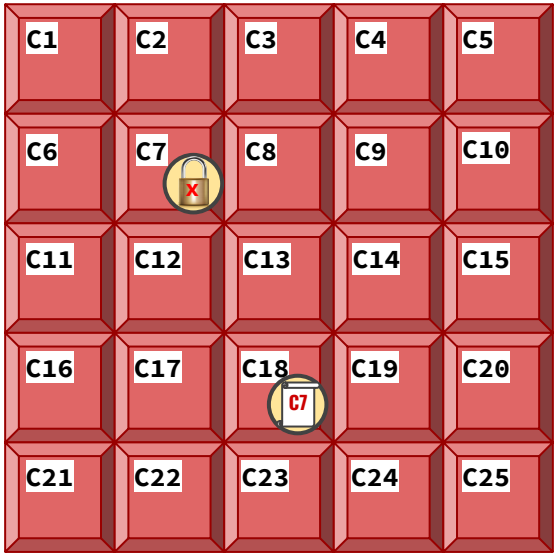
SITUACION 1

- Almacenar la clave **X** en la casilla C7
- Almacenar una copia de la misma clave en la casilla 18



SITUACION 2

- Almacenar la clave **X** en la casilla C7
- Almacenar un mensaje en la casilla 18 con la dirección de la clave **X**



Vista (simplificada) de la memoria principal

```c
char x = 'c';  // Variable tipo char
int y  = 157;  // Variable tipo int
char *px = &x; // Puntero a un char
int *py  = &y; // Puntero a un int
```

Memoria RAM

- Un puntero almacena la dirección de un valor en la memoria principal
- Se trata de un nuevo tipo de variables
- En computadores modernos (64-bits), los punteros ocupan 8 bytes

Indirección

— — —

&: ¡El mismo de scanf!

& añade un nivel de indirección
- &x es la dirección al contenido de la variable x

***** elimina un nivel de indirección
- Si p es un puntero, *p es el contenido apuntado por p

Memoria RAM


```c
int x  = 157;
int *p  = &x;
int y  = *p;
int z  = *p + 3;

printf("y: %d\n", y); // ¿Qué imprime?
printf("z: %d\n", z); // ¿Qué imprime?
```

Ejemplos: ¿Qué imprime?

```c
int a  = 80;
int *b  = &a;
int c  = *b;

printf("c: %d\n", c);
```

Espacio para dibujar


```c
int a  = 80;
int *b  = &a;
*b  += 20;

printf("b: %d\n", *b);
```

Espacio para dibujar

Ejemplos: ¿Qué imprime?

```c
int a  = 80;
int *b  = &a;
int *c  = b;
*c = 11;
printf("b: %d\n", *b);
```

Espacio para dibujar

```c
float a  = 20.3;
float *b  = &a;

printf("%f\n", *b * *b);
```

Espacio para dibujar

Imprimir un puntero

```c
int a  = 80;
int *b  = &a;
long s = sizeof(b);

printf("Puntero b: %p\n", b);
printf("Contenido de b: %d\n", *b);
printf("El tamaño del puntero es: %ld\n", s);
```

¿Por qué cambia el puntero en cada ejecución?

Una ejecución

```
> Puntero b: 0x7ffec5a63b6c
> Contenido de b: 80
> El tamaño del puntero es: 8
```

Otra ejecución

```
> Puntero b: 0x7ffdf5761e2c
> Contenido de b: 80
> El tamaño del puntero es: 8
```

El puntero nulo (NULL)

Un valor NULL indica un puntero vacío o inválido

```c
int *a  = NULL;
printf("Puntero a: %p\n", a);
printf("Contenido a: %d\n", *a);
```

```
> Puntero a: (nil)
> Segmentation fault (core dumped)
```

```c
int *a  = NULL;
printf("Puntero a: %p\n", a);
if(a != NULL)
  printf("Contenido a: %d\n", *a);
else
  printf("Contenido a: NULL\n");
```

```
> Puntero a: (nil)
> Contenido a: NULL
```

Punteros y arreglos

```
int arr[6]  = {3, 5, 1, 0, 2, -2};
```

- arr[0] es equivalente a *arr
- arr[3] es equivalente a *(arr+3)
- &arr[3] es equivalente a (arr+3)

```
int arr[6]  = {3, 5, 1, 0, 2, -2};
int *b = arr;
printf("sizeof(arr): %d - sizeof(b): %d\n",
          (int)sizeof(arr), (int)sizeof(b));
```

Salida

```
> sizeof(arr): 24 - sizeof(b): 8
```

Aritmética de punteros

```c
int a  = 3;
int *p = &a;
printf("a: %d - p: %p\n", a, p);
(*p)++;
printf("a: %d - p: %p\n", a, p);
*p++;
printf("a: %d - p: %p\n", a, p);
p++;
printf("a: %d - p: %p\n", a, p);
```

Salida

```
> a: 3 - p: 0x7fff6d318854
> a: 4 - p: 0x7fff6d318854
> a: 4 - p: 0x7fff6d318858
> a: 4 - p: 0x7fff6d31885c
```

```c
int arr[6]  = {3, 5, 1, 0, 2, -2};
int *b = arr;
*b -= 3;
b += 3;
*b = 9;
for(int i=0; i < 6; i++)
   printf("%d ", arr[i]);
```

Salida

```
0 5 1 9 2 -2
```

¡A practicar!

Ejemplo 1:
punteros.c

Ejemplo 2:
arreglos1.c

Ejemplo 3:
arreglos2.c

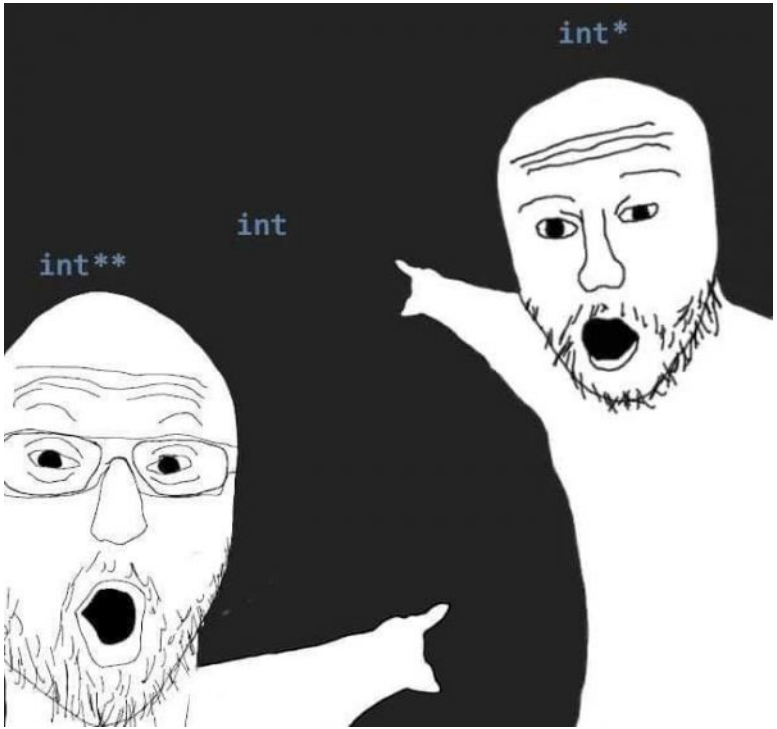
Ejemplo 4:
intercambio.c

Ejemplo 5:
strlen.c

Ejemplo 6:
arreglo_punteros.c

Ejemplo 7:
archivos.c

Si entienden el meme, entendieron la clase :D



URL: https://www.globalnerdy.com/wp-content/uploads/2021/10/pointers-600x565.png