

1

# Apuntaores

# ¿Qué es un apuntador?

2

- Un puntero es un objeto que apunta a otro objeto. Es decir, una variable cuyo valor es la dirección de memoria de otra variable.

Dirección	Etiqueta	Contenido
...		
...		
1502	x	25
1503		
1504		
...		
...		

# Apuntadores

3

- En C no se debe indicar numéricamente la dirección de la memoria, si no que se usa una etiqueta que conocemos como **variable**.



Las direcciones de memoria dependen de la arquitectura del ordenador y de la gestión que el sistema operativo haga de ella.

# ¿Cómo se declaran los apuntadores?

4

- Para declarar un apuntador se especifica el tipo de dato al que apunta, el operador '\*', y el nombre del apuntador.
- Un puntero tiene su propia dirección de memoria.
- La sintaxis es la siguiente:
- <tipo de dato apuntador> \*<identificador del apuntador>

```
int    * punt;  
char   * car;  
float  * num;
```

# ¿Cómo se declaran los apuntadores?

5

- Al igual que el resto de las variables, los apuntadores se enlazan a tipos de datos específicos, de manera que a un apuntador sólo se le puede asignar direcciones de variables del tipo especificado en la declaración

```
int    * punt;  
char   * car;  
float  * num;
```

# Tipos de apuntadores

6

- Hay tantos tipos de apuntadores como tipos de datos.
- Se puede también declarar apuntadores a estructuras más complejas.
  - ▣ Funciones
  - ▣ Struct
  - ▣ Ficheros
- Se pueden declarar punteros vacíos o nulos.

# ¿Qué es la referenciación?

7

- La referenciación es obtener la dirección de una variable.
- Se hace a través del operador '&', aplicado a la variable a la cual se desea saber su dirección

`&x ; //La dirección de la variable`

# ¿Qué es la referenciación?

8

- No hay que confundir una dirección de memoria con el contenido de esa dirección de memoria.

Dirección	Etiqueta	Contenido
...		
...		
1502	x	25
1503		
1504		
...		
...		

`x = 25;` //El contenido de la variable  
`&x = 1502 ;` //La dirección de la variable

# Fragmento de código - referenciación

9

→ **int dato;** //variable que almacenará un carácter.

**int \*punt;** //declaración de puntero a carácter.

**punt = &dato;** //en la variable punt guardamos la dirección  
//de memoria de la variable dato;  
// punt apunta a dato.

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503		
1504		
...		

# Fragmento de código - referenciación

10

```
int dato;           //variable que almacenará un carácter.
```

→ int \*punt; //declaración de puntero a carácter.

```
punt = &dato;      //en la variable punt guardamos la dirección  
                   //de memoria de la variable dato;  
                   // punt apunta a dato.
```

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503	*punt	
1504		
...		

# Fragmento de código - referenciación

11

```
int dato;           //variable que almacenará un carácter.
```

```
int *punt;         //declaración de puntero a carácter.
```

```
→ punt = &dato;    //en la variable punt guardamos la dirección  
                   //de memoria de la variable dato;  
                   // punt apunta a dato.
```

Dirección	Etiqueta	Contenido
...		
...		
1502	dato	/0
1503	*punt	1502
1504		
...		

# ¿Qué es la desreferenciación?

12

- Es la obtención del valor almacenado en el espacio de memoria donde apunta un apuntador.
- Se hace a través del operador “\*”, aplicado al apuntador que contiene la dirección del valor.

`*p ; //El contenido de p`

# Fragmento de código - referenciación

13

```
int x=17, y;  
int * p;  
p = &x;  
printf ("El valor de x es %d", *p);  
y=*p+3;  
printf ("El valor de y es %d", *p);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	
1504		
...		

# Fragmento de código - referenciación

14

```
int x=17, y;  
→ int * p;  
p = &x;  
printf ("El valor de x es %d", *p);  
y=*p+3;  
printf ("El valor de y es %d", *p);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	
1504	*p	
...		

# Fragmento de código - referenciación

15

```
int x=17, y;  
int * p;  
→ p = &x;  
printf ("El valor de x es %d", *p);  
y=*p+3;  
printf ("El valor de y es %d", *p);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	
1504	*p	1502
...		

# Fragmento de código - referenciación

16

```
int x=17, y;  
int * p;  
p = &x;  
→printf ("El valor de x es %d", *p);  
y=*p+3;  
printf ("El valor de y es %d", y);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	
1504	*p	1502
...		

El valor de x es 17

Presione cualquier tecla para continuar...

# Fragmento de código - referenciación

17

```
int x=17, y;  
int * p;  
p = &x;  
printf ("El valor de x es %d", *p);  
→ y= *p+3;  
printf ("El valor de y es %d", y);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	20
1504	*p	1502
...		

El valor de x es 17

Presione cualquier tecla para continuar...

# Fragmento de código - referenciación

18

```
int x=17, y;  
int * p;  
p = &x;  
printf ("El valor de x es %d", *p);  
y=*p+3;  
→ printf ("El valor de y es %d", y);
```

Dirección	Etiqueta	Contenido
...		
...		
1502	x	17
1503	y	20
1504	*p	1502
...		

El valor de x es 17  
El valor de y es 20  
Presione cualquier tecla para continuar...

# Asignación de apuntadores

19

- A un apuntador se pueden asignar direcciones de variables a través del operador de referencia ('&') o direcciones almacenadas en otros apuntadores.

# Direcciones inválidas

20

- Un apuntador puede contener una dirección inválida por:
  - Cuando se declara un apuntador, posee un valor cualquiera que no se puede conocer con antelación.
  - Despues de que ha sido inicializado, la dirección que posee puede dejar de ser válida por que la variable asociada termina su ámbito o por que ese espacio de memoria fue reservado dinámicamente.

# Ejemplo

21

int\*p, y;

```
void func()
{
    int x=40;
    p=&x;
    y=*p;    //Correcto
    *p=23;   //Correcto
}

int main(void)
{
    func();
    y=*p;    //Incorrecto
    *p=25;   //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504		
1505		
1506		
...		
...		

# Ejemplo

22

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

→ int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504		
1505		
1506		
...		
...		

# Ejemplo

23

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504		
1505		
1506		
...		
...		

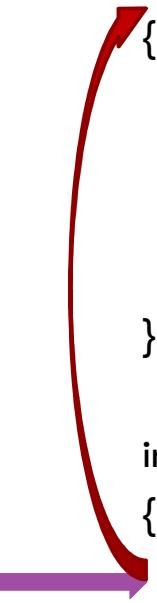
# Ejemplo

24

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504		
1505		
1506		
...		
...		

# Ejemplo

25

```
int*p, y;  
  
→ void func()  
{  
    int x=40;  
    p=&x;  
    y=*p; //Correcto  
    *p=23; //Correcto  
}  
  
int main(void)  
{  
    func();  
    y=*p; //Incorrecto  
    *p=25; //Incorrecto  
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);  
}
```

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504		
1505		
1506		
...		
...		

# Ejemplo

26

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

→

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	
1503	y	
1504	x	40
1505		
1506		
...		
...		

# Ejemplo

27

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

→

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	
1504	x	40
1505		
1506		
...		
...		

# Ejemplo

28

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    → y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	40
1504	x	40
1505		
1506		
...		
...		

# Ejemplo

29

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	40
1504	x	23
1505		
1506		
...		
...		

# Ejemplo

30

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



A diagram illustrating the memory layout and pointer usage. It consists of a table with three columns: Dirección (Address), Etiqueta (Label), and Contenido (Content). The table has 8 rows, with the first row containing ellipses (...). The second row also contains ellipses (...). The third row has address 1502, label \*p, and content 1504. The fourth row has address 1503, label y, and content 40. The fifth row has address 1504, label x, and content 23. The sixth row contains ellipses (...). The seventh row contains ellipses (...). The eighth row contains ellipses (...).

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	40
1504	x	23
1505		
1506		
...		
...		

# Ejemplo

31

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	° 23
1504	x	23
1505		
1506		
...		
...		

# Ejemplo

32

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```



Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	° 23
1504	x	° 25
1505		
1506		
...		
...		

# Ejemplo

33

```
int*p, y;

void func()
{
    int x=40;
    p=&x;
    y=*p; //Correcto
    *p=23; //Correcto
}

int main(void)
{
    func();
    y=*p; //Incorrecto
    *p=25; //Incorrecto
    printf (" El valor de y es %d \nEl valor de *p es %d \n El valor de p es %p", y , *p, p);
}
```

Dirección	Etiqueta	Contenido
...		
...		
1502	*p	1504
1503	y	° 23
1504	x	° 25
1505		
1506		
...		
...		

El valor de y es 23  
El valor de \*p es 25  
El valor de p es 001244FF  
Presione cualquier tecla para continuar...

# La dirección NULL

34

- Cuando no se desea que el apuntador apunte a algo, se le suele asignar el valor de NULL, en cuyo caso se dice que el apuntador es nulo (no apunta a nada).
- NULL es una macro típicamente definida en archivos de cabecera como stddef.h y stdlib.h.
- Se utiliza para proporcionar a un programa un medio de conocer cuándo un apuntador contiene una dirección inválida.

# Apuntadores a apuntadores

35

- Dado que un apuntador es una variable que apunta a otra, fácilmente se puede deducir que pueden existir apuntadores a apuntadores, y a su vez los segundos pueden apuntar a apuntadores.

```
char c = 'z';
char *pc = &c;
char **ppc = &pc;
char ***pppc = &ppc;

***pppc = 'm'
```

Dirección	Etiqueta	Contenido
...		
1502	c	z
1503	pc	1502
1504	ppc	1503
1505	pppc	1504
...		
...		

# Apuntadores constantes

36

- Es posible declarar apuntadores a constantes. De esta manera, no se permite la modificación de la dirección almacenada en el apuntador, pero si se permite la modificación del valor al que apunta.

```
int x = 5, y = 7;  
int *const p = &x;  
*p=3;  
p=&y;
```

Dirección	Etiqueta	Contenido
...		
1502	x	5
1503	y	7
1504	*p	1502
1505		
...		
...		

# Paso de parámetros por referencia

37

- En este tipo de llamadas los argumentos contienen direcciones de variables.
- Dentro de la función la dirección se utiliza para acceder al argumento real.
- En las llamadas por referencia cualquier cambio en la función tiene efecto sobre la variable cuya dirección se pasó como argumento.

# Paso de parámetros por referencia

38

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503		
1504		
1505		
...		
...		

# Paso de parámetros por referencia

39

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504		
1505		
...		
...		

# Paso de parámetros por referencia

40

```
int main(void)
{
    int x = 2;
    int y = 5;
    → printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}
```

```
void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504		
1505		
...		
...		

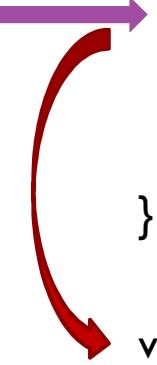
Antes x = 2 , y = 5

# Paso de parámetros por referencia

41

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```



Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504		
1505		
...		
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

42

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}
```

→ void intercambio(int \*a, int \*b){  
 int temp;  
 temp = \*b;  
 \*b = \*a;  
 \*a = temp;  
}

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504	*a	1502
1505	*b	1503
...		
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

43

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504	*a	1502
1505	*b	1503
1506	temp	
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

44

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	5
1504	*a	1502
1505	*b	1503
1506	temp	5
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

45

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```



Dirección	Etiqueta	Contenido
...		
1502	x	2
1503	y	2
1504	*a	1502
1505	*b	1503
1506	temp	5
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

46

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	5
1503	y	2
1504	*a	1502
1505	*b	1503
1506	temp	5
...		

Antes x = 2 , y = 5

# Paso de parámetros por referencia

47

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}

void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```



Antes x = 2 , y = 5

Dirección	Etiqueta	Contenido
...		
1502	x	5
1503	y	2
1504	*a	1502
1505	*b	1503
1506	temp	5
...		

# Paso de parámetros por referencia

48

```
int main(void)
{
    int x = 2;
    int y = 5;
    printf ("Antes x = %d, y = %d \n", x, y);
    intercambio (&x, &y);
    printf ("Despues x = %d, y = %d \n", x, y);
    system("Pause");
}
```

```
void intercambio(int *a, int *b){
    int temp;
    temp = *b;
    *b = *a;
    *a = temp;
}
```

Dirección	Etiqueta	Contenido
...		
1502	x	5
1503	y	2
1504	*a	1502
1505	*b	1503
1506	temp	5
...		

Antes x = 2 , y = 5

Despues x = 5, y = 2

Presione cualquier tecla para continuar...

# La función sizeof()

49

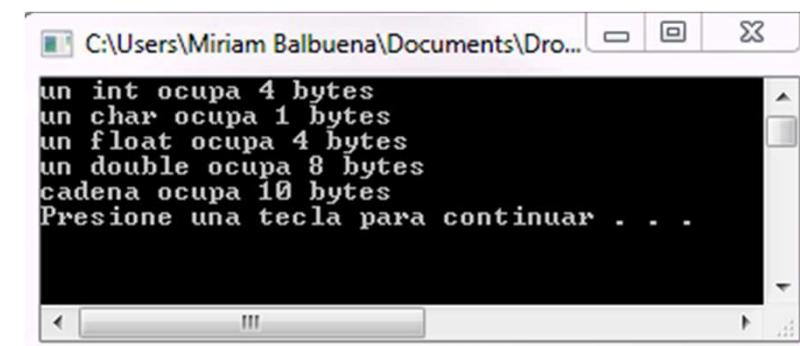
- Devuelve el tamaño en bytes que ocupa un tipo o variable en memoria.

```
char cadena [10];
printf ("un int ocupa %d bytes", sizeof(int));
printf ("un char ocupa %d bytes", sizeof(char));
printf ("un float ocupa %d bytes", sizeof(float));
printf ("un double ocupa %d bytes", sizeof(double));
printf ("cadena ocupa %d bytes", sizeof(cadena));
```

# La función sizeof()

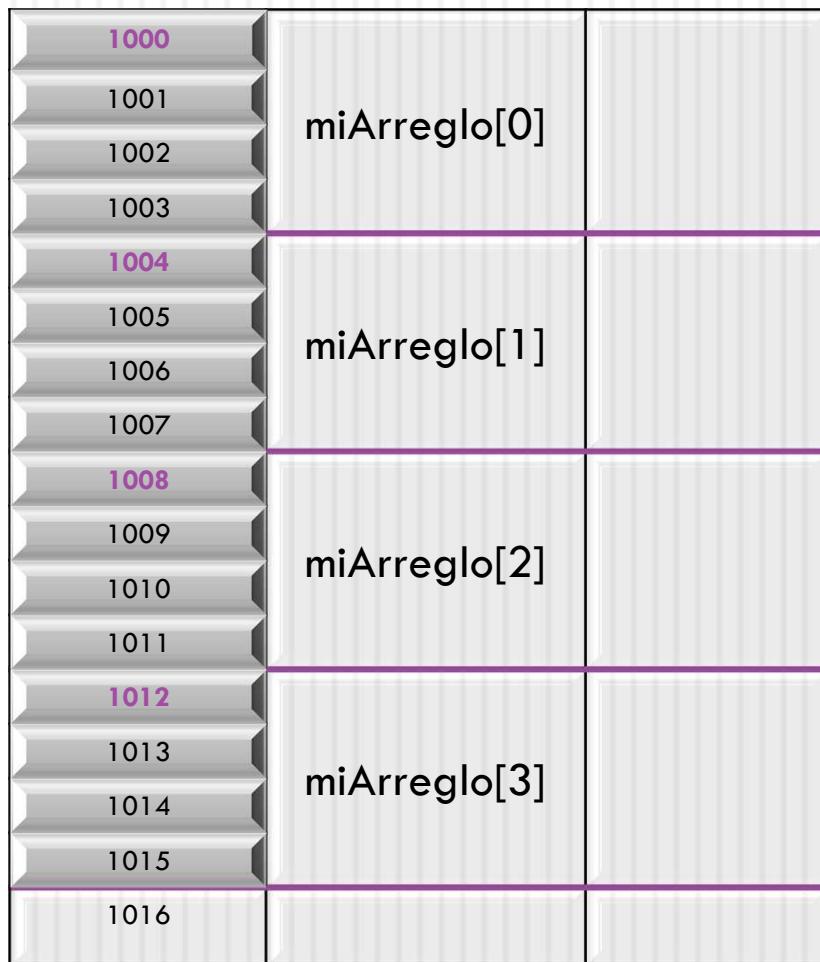
50

Dirección	Contenido
1000	
1001	
1002	<b>int</b>
1003	
1004	<b>char</b>
1005	
1006	
1007	<b>float</b>
1008	
1009	
1010	
1011	
1012	
1013	<b>double</b>
1014	
1015	
1016	



C:\Users\Miriam Balbuena\Documents\Dropbox\... un int ocupa 4 bytes un char ocupa 1 bytes un float ocupa 4 bytes un double ocupa 8 bytes cadena ocupa 10 bytes Presione una tecla para continuar . . .

- `int miArreglo[4] = {1,2,3,4}`



`sizeof(miArreglo)`

16



□ int miArreglo[4] = {1,2,3,4}

1000	miArreglo[0]	
1001		
1002		
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016		

sizeof(int)

4

?

i



- `int miArreglo[4] = {1,2,3,4}`

1000	miArreglo[0]	
1001		
1002		
1003		
1004		
1005	miArreglo[1]	
1006		
1007		
1008		
1009	miArreglo[2]	
1010		
1011		
1012		
1013	miArreglo[3]	
1014		
1015		
1016		

`sizeof(miArreglo)`

`sizeof(int)`



# Ejemplo

54

```
int main()
{
    int array[10] = {1,2,3,4,5,6,7,8,9,0};
    int len=sizeof(array)/sizeof(int);
    printf ("Los bytes del arreglo son: %d\n", sizeof(array));
    printf ("Cada entero tiene: %d bytes\n", sizeof(int));
    printf ("El arreglo tiene %d elementos\n", len);
    system("pause");
    return 0;
}
```



Los bytes del arreglo son: 40  
Cada entero tiene: 4 bytes  
El arreglo tiene 10 elementos  
Presione una tecla para continuar . . .

# Asignación dinámica de memoria

55

- Los programas pueden crear variables globales o locales.
- Las variables declaradas globales en sus programas se almacenan en posiciones fijas de memoria (segmento de datos) y todas las funciones pueden utilizar estas variables.
- Las variables locales se almacenan en la pila (stack) y existen solo mientras están activas las funciones donde están declaradas.
- En ambos casos el espacio de almacenamiento se reserva en el momento de compilación del programa.

# Asignación dinámica de memoria

56

- Para asignar memoria dinámicamente se utilizan las funciones `malloc()` y `free()`, definidas típicamente en el archivo `stdlib.h`.



# free()

57

- La función free() permite liberar la memoria reservada a través de un apuntador.

**void free (void\* ptr);**

**ptr** es un puntero de cualquier tipo que apunta a un área de memoria reservada previamente con **malloc**.

# malloc()

58

- La función malloc() reserva memoria y retorna su dirección, o retorna NULL en caso de no haber conseguido suficiente memoria.

```
Void *malloc(size_t tam_bloque)
```

- malloc() reserva memoria sin importar el tipo de datos que almacenará en ella.

# Ejemplo

59

```
int main(void)
{
    int i,n;
    char * buffer;

    printf (" Teclea la longitud de la cadena? ");
    scanf ("%d", &i);

    buffer = (char*) malloc (i+1);
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	
1503	n	
1504		
1505		
1506		
...		

# Ejemplo

60

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	
1503	n	
1504	*buffer	
1505		
1506		
...		

# Ejemplo

Teclea la longitud de la cadena

61

```
int main(void)
{
    int i,n;
    char * buffer;

    → printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	
1503	n	
1504	*buffer	
1505		
1506		
...		

# Ejemplo

Teclea la longitud de la cadena

62

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	
1504	*buffer	
1505		
1506		
...		

# Ejemplo

Teclea la longitud de la cadena

63

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    → buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	
1504	buffer[0]	
1505	buffer[1]	
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

64

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    ➔ if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	
1504	buffer[0]	
1505	buffer[1]	
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

65

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	0
1504	buffer[0]	
1505	buffer[1]	
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

66

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	0
1504	buffer[0]	f
1505	buffer[1]	
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

67

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	1
1504	buffer[0]	f
1505	buffer[1]	
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

68

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	1
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

69

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	2
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

70

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	2
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

71

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	3
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

72

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	3
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

73

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

74

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	p
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

75

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    → for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	p
1509	buffer[5]	
...		
...		

# Ejemplo

Teclea la longitud de la cadena

76

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	p
1509	buffer[5]	\0
...		
...		

# Ejemplo

Teclea la longitud de la cadena 5  
Random string: fmabp

77

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    → printf ("Random string: %s\n", buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	p
1509	buffer[5]	\0
...		
...		

# Ejemplo

78

Teclea la longitud de la cadena 5  
Random string: fmabp  
Presione cualquier tecla para continuar...

```
int main(void)
{
    int i,n;
    char * buffer;

    printf ("Teclea la longitud de la cadena");
    scanf ("%d", &i);

    buffer = (char*) malloc ((i+1)*sizeof(char));
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Random string: %s\n",buffer);
    free (buffer);
    system("Pause");
}
```

Dirección	Etiqueta	Contenido
...		
1502	i	5
1503	n	4
1504	buffer[0]	f
1505	buffer[1]	m
1506	buffer[2]	a
1507	buffer[3]	b
1508	buffer[4]	p
1509	buffer[5]	\0
...		
...		

- Crea un arreglo entero de tamaño  $x$ , en donde  $x$  es ingresado por teclado.
- Llena todos los elementos del arreglo con datos ingresados por el usuario.
- Muestra los valores



# Apuntadores a arreglos

80

- El nombre de un arreglo es simplemente un apuntador constante al inicio del arreglo

```
int lista_arr[3]={10,20,30};  
int *lista_ptr;  
lista_ptr = lista_arr;
```



Dirección	Etiqueta	Contenido
...		
1502	arr[0]	10
1503	arr[1]	20
1504	arr[2]	30
1505	*list_arr	1502
...		
...		



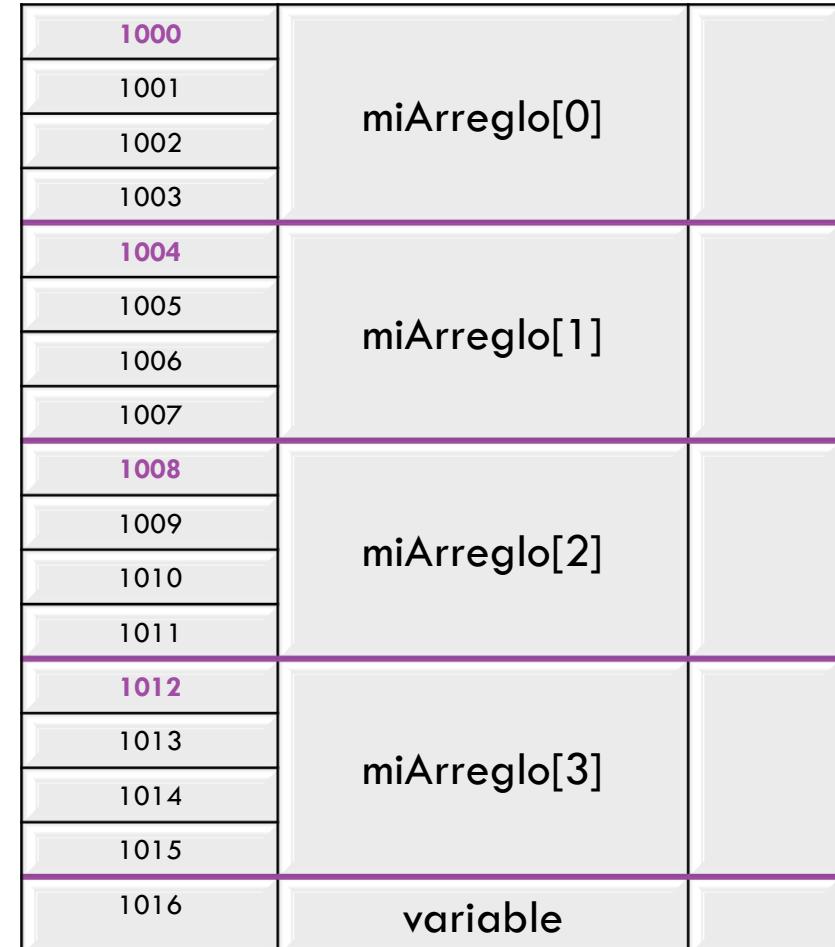
Se crea el apuntador lista\_ptr para poder  
modificando la dirección a donde apunta

# Arreglos

RAM

81

- `miArreglo = 1000`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`



# Direcciones del arreglo

82

```
int i[10], x;
float f[10];

int main(void)
{
    printf("\t\tEntero\t\tFlotante\n\n");
    for(x=0; x<10; x++) {
        printf("Elemento %d:\t%d\t\t%d\n", x, &i[x], &f[x]);
    }

    system("Pause");
}
```

	Entero	Flotante
Elemento 0:	4210784	4210832
Elemento 1:	4210788	4210836
Elemento 2:	4210792	4210840
Elemento 3:	4210796	4210844
Elemento 4:	4210800	4210848
Elemento 5:	4210804	4210852
Elemento 6:	4210808	4210856
Elemento 7:	4210812	4210860
Elemento 8:	4210816	4210864
Elemento 9:	4210820	4210868

Presione una tecla para continuar . . .

## Aritmética de operadores



# Incremento de operadores

84

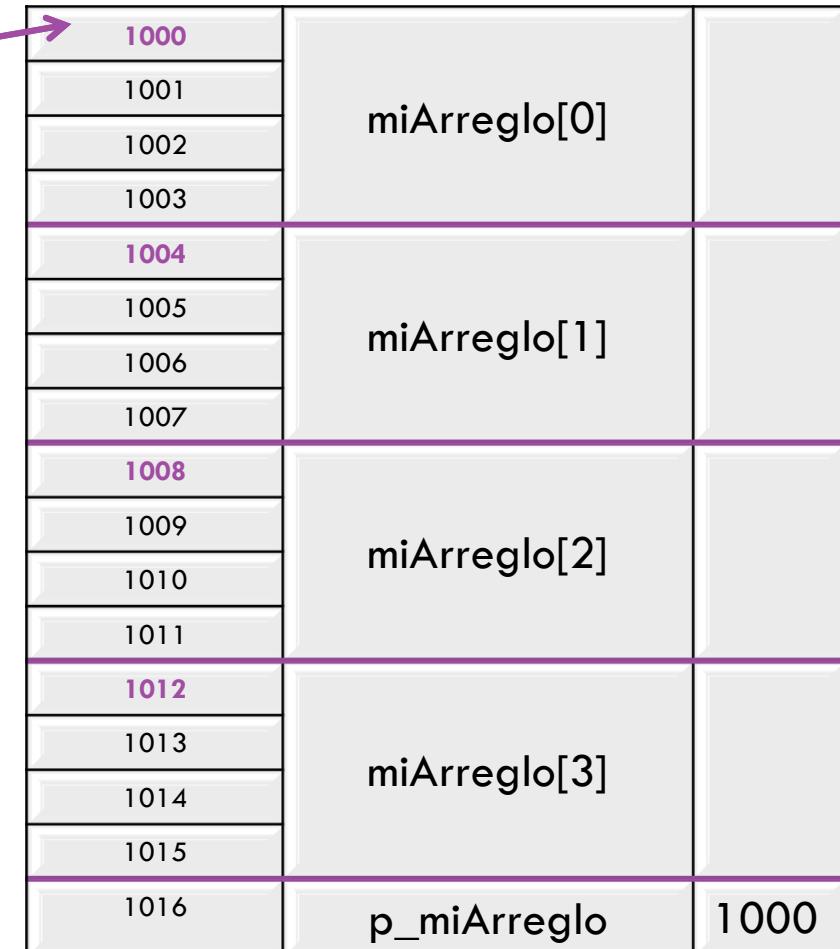
- Cuando se incrementa un apuntador se está incrementando su valor.
  
- Si incrementamos en 1 el valor del apuntador, C sabe el tipo de dato al que apunta e incrementa la dirección guardada en el apuntador en el tamaño del tipo de dato.

# Arreglos

RAM

85

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`



# Arreglos

RAM

86

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo ++;`

¿Ahora que valor tiene `p_miArreglo`?



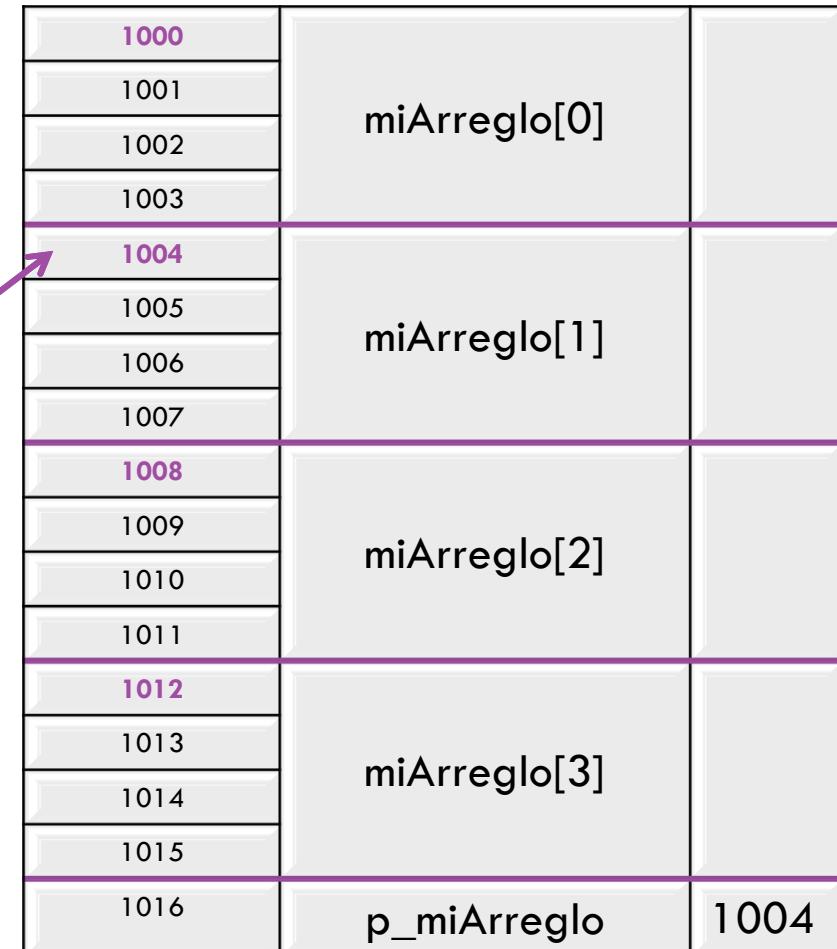
1000	miArreglo[0]	
1001		
1002		
1003		
1004	miArreglo[1]	
1005		
1006		
1007		
1008	miArreglo[2]	
1009		
1010		
1011		
1012	miArreglo[3]	
1013		
1014		
1015		
1016	p_miArreglo	1004

# Arreglos

RAM

87

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo ++;`

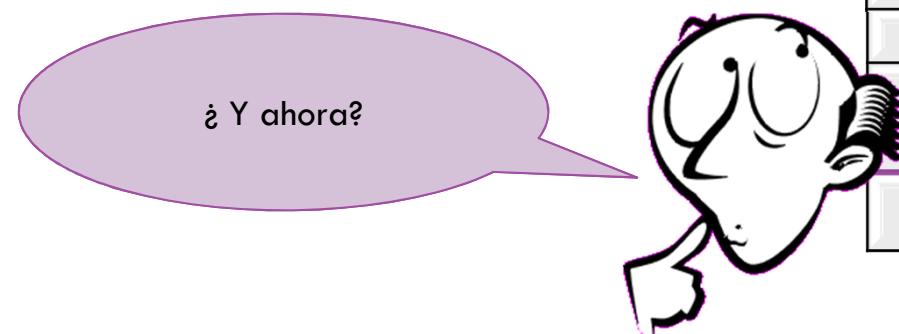


# Arreglos

RAM

88

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo += 2;`



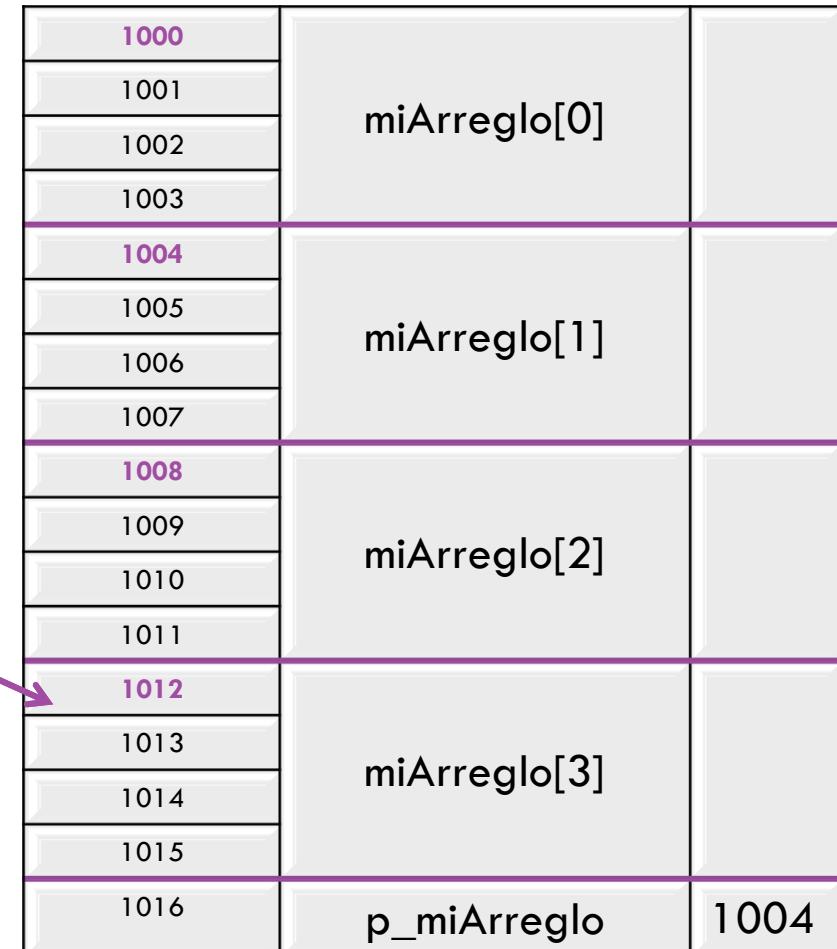
1000	miArreglo[0]	
1001		
1002		
1003		
1004	miArreglo[1]	
1005		
1006		
1007		
1008	miArreglo[2]	
1009		
1010		
1011		
1012	miArreglo[3]	
1013		
1014		
1015		
1016	p_miArreglo	1004

# Arreglos

RAM

89

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo += 2;`

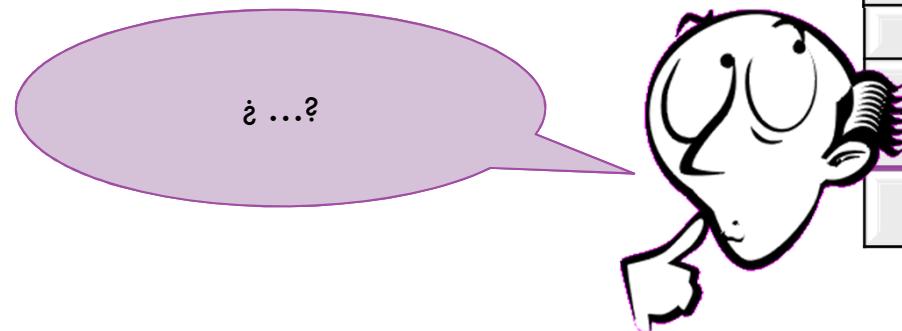


# Arreglos

RAM

90

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo --;`



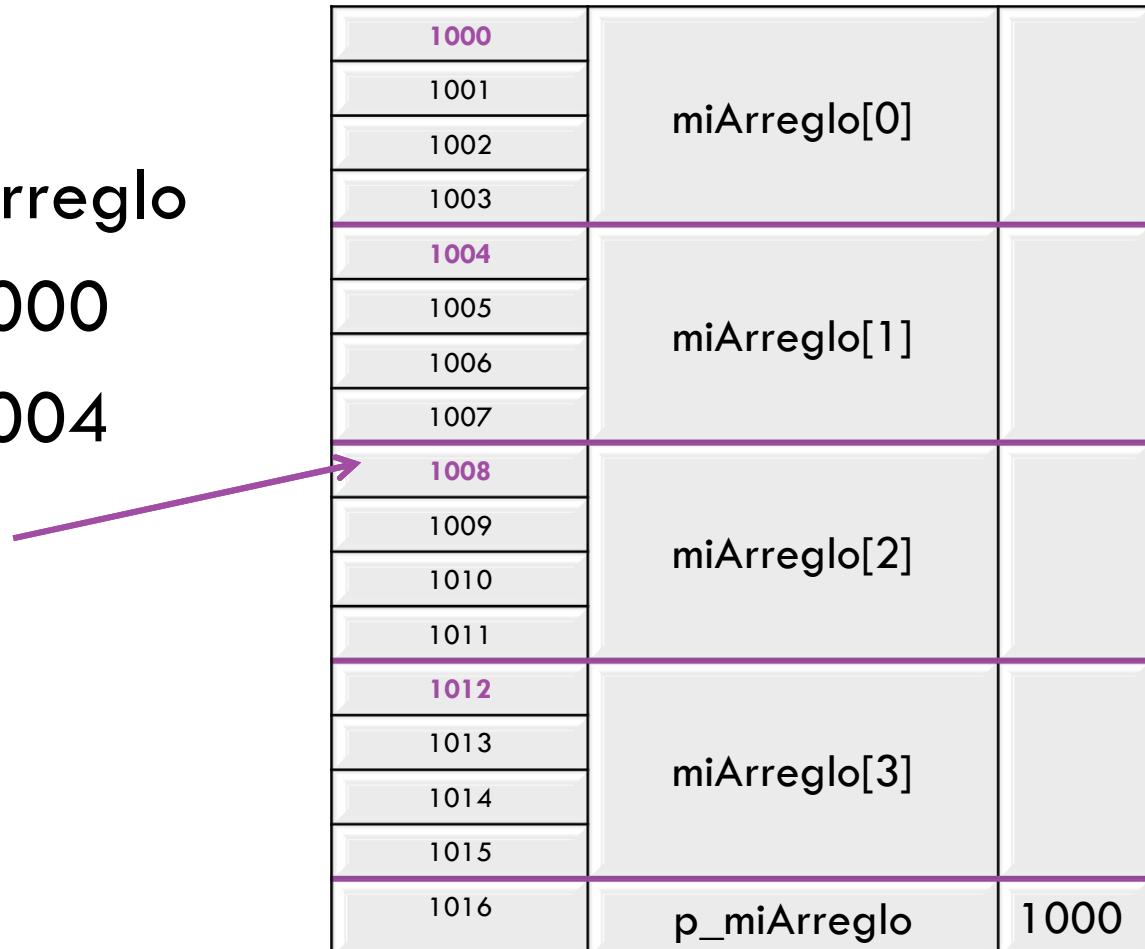
1000	miArreglo[0]	
1001		
1002		
1003		
1004	miArreglo[1]	
1005		
1006		
1007		
1008	miArreglo[2]	
1009		
1010		
1011		
1012	miArreglo[3]	
1013		
1014		
1015		
1016	p_miArreglo	1000

# Arreglos

RAM

91

- `miArreglo = 1000`
- `p_miArreglo = miArreglo`
- `&miArreglo[0] = 1000`
- `&miArreglo[1] = 1004`
- `p_miArreglo --;`



- Crea un arreglo entero de tamaño x, en donde x es ingresado por teclado.
  
- Llena todos los elementos del arreglo con datos ingresados por el usuario usando apuntadores.

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	
1008		
1012		
1016		
1020		
1024		
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	
1008	*buffer	
1012	*p_buffer	
1016		
1020		
1024		
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    →printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	
1008	*buffer	
1012	*p_buffer	
1016		
1020		
1024		
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	3
1008	*buffer	
1012	*p_buffer	
1016		
1020		
1024		
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    ➔ buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	3
1008	*buffer	1016
1012	*p_buffer	
1016	buffer[0]	
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	3
1008	*buffer	1016
1012	*p_buffer	
1016	buffer[0]	
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    →p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	0
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	0
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++); →
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	0
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	5
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++); →
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	0
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	1
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	1
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++); →
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	1
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++); →
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	1
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	2
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	2
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);

    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	2
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1028
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1028
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    →p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	0
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	0
1008	*buffer	1016
1012	*p_buffer	1016
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	0
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	1
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	1
1008	*buffer	1016
1012	*p_buffer	1020
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	1
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	2
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	2
1008	*buffer	1016
1012	*p_buffer	1024
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	2
1008	*buffer	1016
1012	*p_buffer	1028
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```



1000	i	3
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1028
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

# Ejemplo

```

int main(void)
{
    int i,n;
    int * buffer, * p_buffer;

    printf ("Teclea la longitud del arreglo");
    scanf ("%d", &n);

    buffer = (int*) malloc((n)* sizeof(int));
    if (buffer==NULL) exit (1);

    p_buffer = buffer;
    for (i=0;i<n; i++){
        printf("Ingresa el valor %d \n", i);
        scanf("%d", p_buffer++);
    }

    p_buffer = buffer;
    printf("\nLos valores son\n");
    for (n=0; n<i; n++){
        printf("arreglo[%d] = %d \n", n, *p_buffer++);
    }

    free (buffer);
    system("Pause");
}

```

1000	i	3
1004	n	3
1008	*buffer	1016
1012	*p_buffer	1028
1016	buffer[0]	5
1020	buffer[1]	6
1024	buffer[2]	7
1028		
1032		
1036		
1040		
1044		
1048		
1013		
1014		
1015		
1016		

`getchar();`

125

- Crea un arreglo de tipo char de tamaño x, en donde x es ingresado por teclado.
- Llena elemento por elemento del arreglo con letras ingresados por el usuario.
- Muestra el arreglo impreso en forma inversa.
- Todo debe ser manejado con apuntadores.

