Estructuras de datos dinámicas I

1ºDAM IoT

Direcciones de memoria y contenido

Ya hemos visto que cuando declaramos una variable lo que se hace internamente es reservar un espacio de memoria. Este espacio de memoria tiene una dirección de memoria para poder acceder a la variable.

int x = 25;

Dirección	1500	1504	1508	1512	**1516**	1520	1524	1528	1532	1536
Contenido	…	…	…	…	**25**	…	…	…	…	…

La dirección de la variable x es 1516.

El contenido de la variable x es 25.

En este ejemplo, he declarado la variable y asignado un valor a la vez. Recuerda que cuando declaro una variable en ese espacio puede haber cualquier cosa (basura).

Punteros

Los punteros:

- Son una de las poderosas herramientas que ofrece el lenguaje C a los programadores.
- Sin embargo, son también una de las más peligrosas, el uso de punteros sin inicializar, etc., es una fuente frecuente de errores en los programas de C, y además, suele producir fallos muy difíciles de localizar y depurar.

Un **puntero es una variable que contiene una dirección de memoria**. Normalmente esa dirección es una posición de memoria de otra variable, por lo cual se suele decir que el puntero “apunta” a la otra variable.

Punteros

Como el contenido de un puntero es una dirección de memoria, el valor contenido puede variar según la naturaleza de la máquina. Siempre será un valor entero sin signo (pues las direcciones de la memoria son 0, 1, 2...).

Punteros - Declaración

La sintaxis de la declaración de una variable puntero es:

tipo *nombre;

El tipo base de la declaración sirve para conocer el tipo de datos al que pertenece la variable a la cual apunta la variable de tipo puntero.

Es decir, un puntero debe tener el mismo tipo de dato del valor al que apuntará.

Algunos ejemplos de declaración de variables puntero son:

int *a; //Apuntará a un variable de tipo int

char *p; //Apuntará a una variable de tipo char

En el momento de la declaración no está apuntando a ningún sitio.

Punteros asignación

El operador & u operador de asignación, es un operador unario que retorna la dirección de su operando.

int x = 25;

int *xPuntero;

xPuntero = &x; //Con esta operación le digo que el puntero apunta a la variable

					x					**xPuntero**
Dirección	1500	1504	1508	1512	**1516**	1520	1524	1528	1532	**1536**
Contenido	…	…	…	…	**25**	…	…	…	…	**1516**

Punteros asignación

El operador **&** u operador de asignación, es un operador unario que retorna la dirección de su operando.

int x = 25; //Un entero ocupa 4 bytes

int *xPuntero;

xPuntero = &x;

Imprimir punteros

Si queremos imprimir punteros tenemos dos opciones dependiendo de lo que queramos hacer:

Imprime la dirección de memoria dónde se guarda el puntero. “**%p**”

printf("Dirección de memoria **%p**\n", **xPuntero**);

Si queremos imprimir el contenido del puntero indicamos el formateo del tipo de dato que guarda:

printf("**%d**\n", ***xPuntero**); //Almacena un entero

printf("**%g**\n", ***yPuntero**); //Almacena un float

Imprimir punteros

Ejemplo impresión de un puntero. Como puedes ver imprime la dirección que tiene guardada, que es la dirección de memoria de la variable Dato.

```
int Dato = 5, *PtrDato;

PtrDato = &Dato;

printf("%p\n", PtrDato);
```

Declara e inicializa una variable de tipo int, float y char.
Declara e inicializa un puntero para cada una de las variables.
Imprime la dirección de memoria guardada por el puntero y el valor de la dirección a la que apunta.
Imprime también la dirección de memoria de las variables de tipo int, float y char. Para ello tendrás que anteponer & al nombre de la variable.
¿Es la misma dirección de los punteros?
Compara la dirección de memoria de tus variables y punteros con las de tus compañeros.

Ejemplo de punteros

Mis direcciones de memoria posiblemente sean diferentes a las tuyas.

```
D. de memoria variable: 0061FF10
Valor apuntado: 0061FF10
Contenido del valor apuntado: 25
---------------
D. de memoria variable: 0061FF0C
Valor apuntado: 0061FF0C
Contenido del valor apuntado: 10.4
---------------
D. de memoria variable: 0061FF0B
Valor apuntado: 0061FF0B
Contenido del valor apuntado: z
```

```c
#include <stdio.h>

int main()
{
    int x = 25;
    int *xPuntero;
    xPuntero = &x;

    printf("D. de memoria variable: %p\n", &x);
    printf("Valor apuntado: %p\n", xPuntero);
    printf("Contenido del valor apuntado: %d\n", *xPuntero);
    printf("---------------\n");

    float y = 10.4;
    float *yPuntero;
    yPuntero = &y;

    printf("D. de memoria variable: %p\n", &y);
    printf("Valor apuntado: %p\n", yPuntero);
    printf("Contenido del valor apuntado: %g\n", *yPuntero);
    printf("---------------\n");

    char z = 'z';
    char *zPuntero;
    zPuntero = &z;

    printf("D. de memoria variable: %p\n", &z);
    printf("Valor apuntado: %p\n", zPuntero);
    printf("Contenido del valor apuntado: %c\n", *zPuntero);
    return 0;
}
```

Punteros asignación

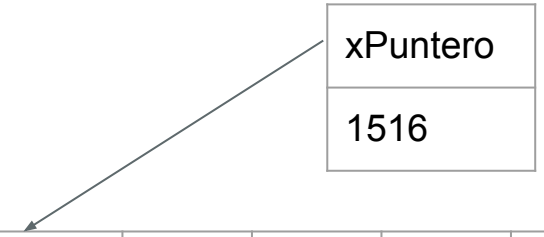
Para asignar un valor a un puntero lo podemos hacer directamente. Para ello debemos anteponer el * al identificador del puntero. Para decirle que guarde el valor en la dirección de memoria a la cual está apuntando.

int x = 25;

int *xPuntero;

xPuntero = &x;

*xPuntero = 1500;



Dirección	1500	1504	1508	1512	**1516**	1520	1524	1528	1532	1536
Contenido	…	…	…	…	**1500**	…	…	…	…	…

Punteros asignación

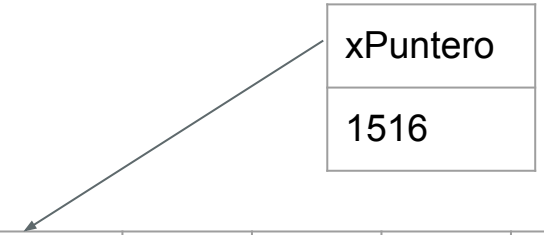
¡Ojo! Si modificas el valor apuntado por el puntero también se modifica la variable original.

int x = 25;

int *xPuntero;

xPuntero = &x;

*xPuntero = 1500;



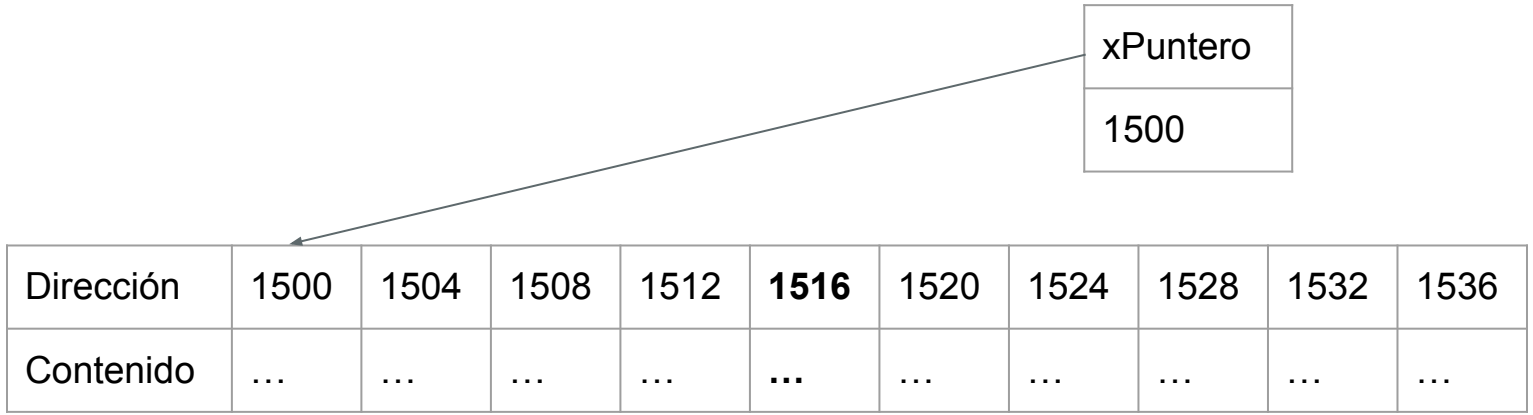
Dirección	1500	1504	1508	1512	**1516**	1520	1524	1528	1532	1536
Contenido	…	…	…	…	**1500**	…	…	…	…	…

Punteros asignación

¡Cuidado! Si no pones * lo que estamos haciendo es modificar la dirección de memoria que guarda.

int *xPuntero;

xPuntero = 1500;



¿Qué imprime el siguiente programa?

Intenta resolverlo en papel primero y luego si quieres comprueba tu resultado en Visual Studio Code.

```c
#include <stdio.h>

int main()
{
    int a, b, c, *p1, *p2;
    p1 = &a;
    *p1 = 1;
    p2 = &b;
    *p2 = 2;
    p1 = p2;
    *p1 = 0;
    p2 = &c;
    *p2 = 3;
    printf("%d %d %d\n", a, b, c);
    return 0;
}
```