




17/08/21

Intro. al manejo dinámico de datos

Continuación





Agenda



01

**Indirección
simple**

02

**Indirección
múltiple**

03

**Repaso
estructuras**



04


**Repaso
uniones**

05

**Repaso
clases**

06

**Ejercicio
mixto**



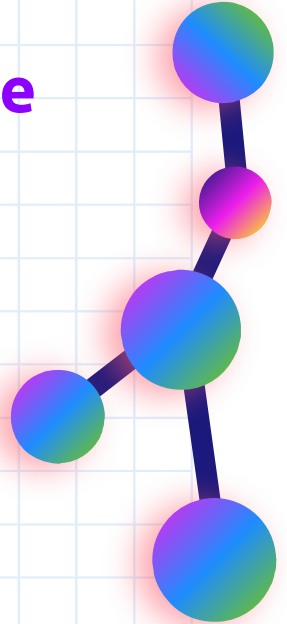
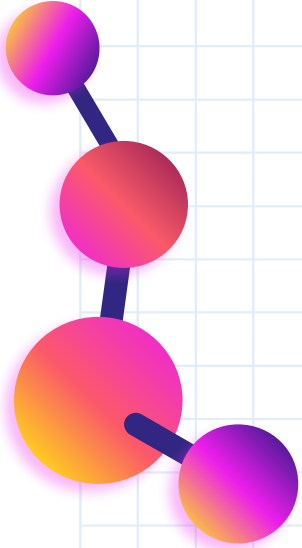
¿Qué es un puntero?

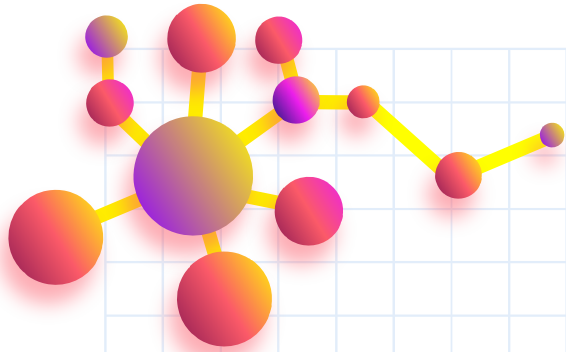
Un puntero es una **variable** que contiene una dirección de memoria

Una variable **int** almacena números enteros.

Una variable **char** almacena caracteres ASCII.

Un **puntero** almacena direcciones de memoria.

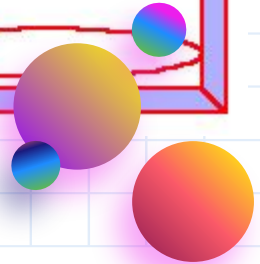
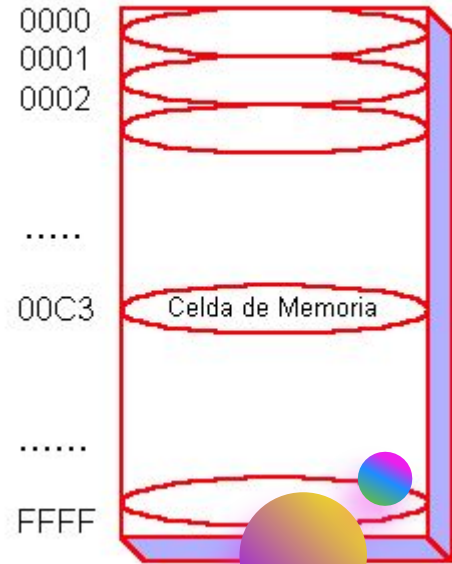




¿Qué es una dirección de memoria?

La memoria de una computadora está dividida en muchos segmentos, que están numerados de manera **secuencial**.

No hay razón por la cual deberíamos preocuparnos por comprender el valor numérico de la dirección de una variable. Lo que nos interesa es saber que **cada celda** tiene una **dirección asignada**.



¿Cómo se obtiene la dirección de memoria de una variable?

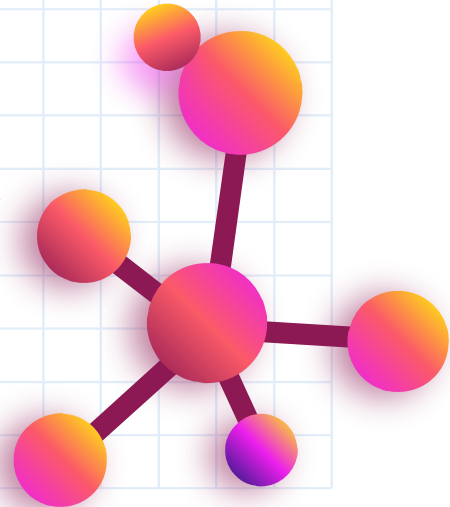


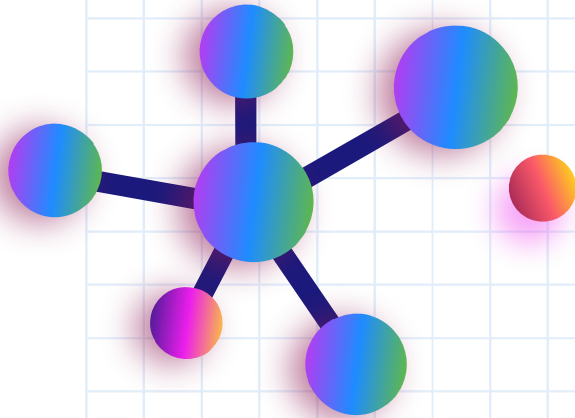
Se utiliza el operador & (ámpersand).

Si se **antepone** a una variable, entonces se obtiene su dirección de memoria.

Un ejemplo de dirección de memoria es **0x7fd7d47dbc**. Este gran número es específico de cada computadora y puede cambiar **cada vez** que se ejecuta un programa.

El resultado que usted ejecute será distinto... **¡prueba!**





Usos del operador * (asterisco)

MULTIPLICACIÓN

(Si se utiliza en una operación)

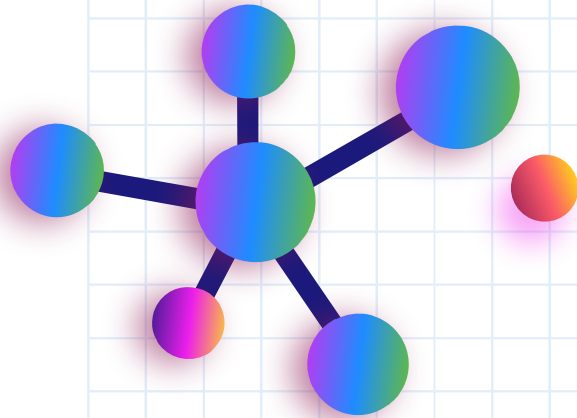
```
int cantidad = 3 * 5;
```

INDIRECCIÓN

(Si se utiliza cuando se declara un puntero)

```
int *puntero;
```

Indica que dicha variable se utilizará como un puntero (y no como una variable normal), es decir, su uso será almacenar la dirección en memoria de otra variable.



Usos del operador * (asterisco)

DESREFERENCIA
(Si se utiliza junto a un puntero ya declarado)

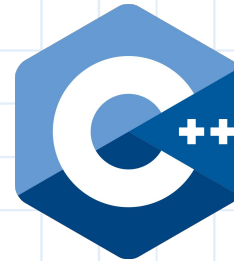
```
cout << (*puntero) << endl;
```

Lee/consulta

```
*puntero = 8;
```

Escribe/asigna

Devuelve el valor contenido (o asigna uno nuevo) en la dirección apuntada por dicho puntero.



```
#include <iostream>
using namespace std;
```



```
void calculos(int num, int *doble, int *triple){
    *doble = 2 * num;
    *triple = 3 * num;
}
```

```
int main(){
    int numero = 2, doble = 0, triple = 0;

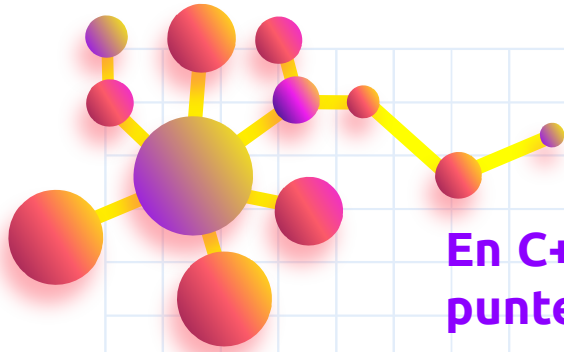
    calculos(numero, &doble, &triple);

    cout << "El doble de 2 es " << doble << endl;
    cout << "El triple de 2 es " << triple << endl;

    return 0;
}
```

Ejemplo

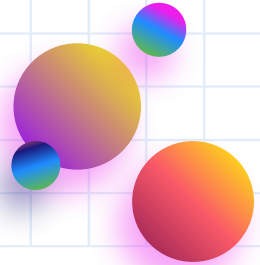
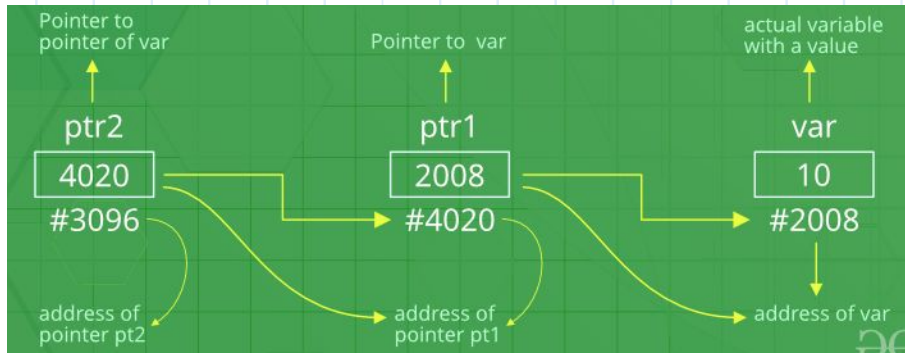




Indirección múltiple

En C++, es posible hacer que un puntero apunte a otro puntero. A esto se le llama **indirección múltiple**.

Cuando un puntero apunta a otro puntero, el primer puntero contiene la dirección del segundo puntero, el cual apunta a la posición que contiene el objeto.



INDIRECCIÓN

Para declarar un puntero a un puntero se coloca un asterisco adicional delante del nombre del puntero.

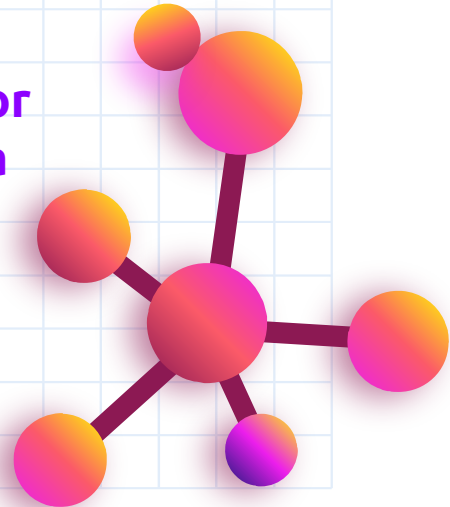
```
char **doble = ...
```

DESREFERENCIA

No se obtendrá el valor original, sino un puntero al valor original. Dicho puntero deberá ser desreferenciado una vez más para poder obtener el valor original

```
**doble = 'A';
```

```
char *nuevo = *doble;  
char letra = *nuevo;  
cout << letra << endl;
```

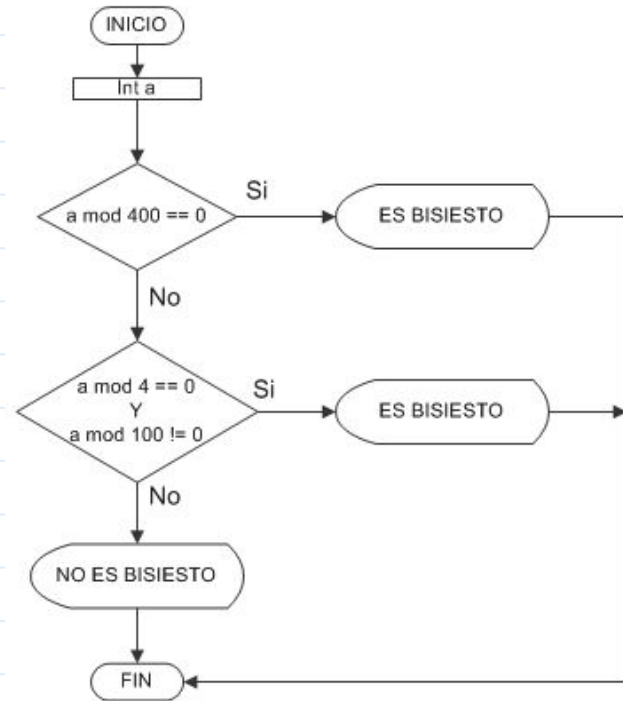


Ejemplo A

Cálculo de año bisiesto

Año bisiesto es el divisible entre 4, salvo que sea año secular (último de cada siglo, terminado en "00"), en cuyo caso también ha de ser divisible entre 400.

De esta forma, los años 1800 y 1900 pese a ser divisibles por 4, no lo son por 400, por lo que fueron años comunes. Por su parte, el año 2000 es divisible tanto por 4 como por 400, por lo tanto sí fue un año bisiesto.





```
7- int main(){
8   int anio = 0;
9   cout << "Digite el anio: ";
10  cin >> anio;
11
12  if(etapa1(&anio))
13      cout << "Si es bisiestro." << endl;
14  else
15      cout << "No es bisiestro." << endl;
16
17  return 0;
18 }
19
20 bool etapa1(int *s){
21     if(*s%400 == 0)
22         return true;
23     else
24         return etapa2(&s);
25 }
26
27 bool etapa2(int **d){
28     if(**d%4 == 0 && **d%100 != 0)
29         return true;
30     else
31         return false;
32 }
```

Variable
común

Enviar dire-
cción de Anio

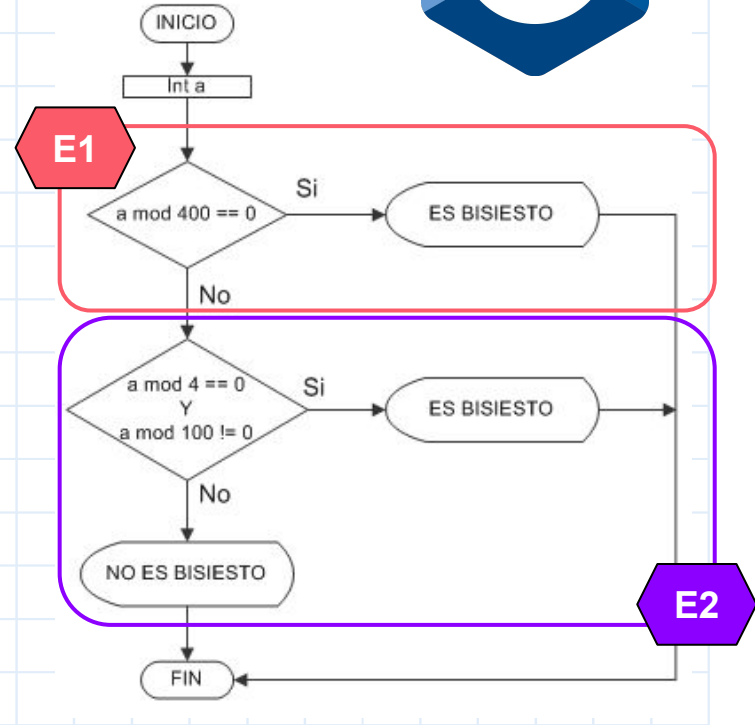
Indirección
simple

Enviar
dirección de S

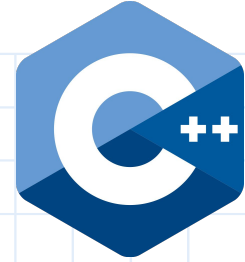
Desreferencia
simple

Indirección
doble

Desreferencia
doble



Ejemplo B

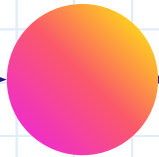


**Inicio del
partido**



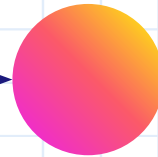
Main:
Se definen
todos los
datos

**90
minutos**



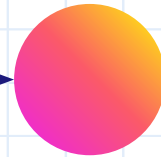
Etapa 1:
Si nadie gana,
entonces ir a
etapa 2

**Tiempo
extra**



Etapa 2:
Si nadie gana,
entonces ir a
etapa 3

**Tiros
penales**



Etapa 3:
Se decide
ganador

```

8 int main(){
9     int puntajeA = 0, puntajeB = 0;
10
11     if(etapa1(&puntajeA, &puntajeB)==1){
12         cout << "Equipo A." << endl;
13         cout << puntajeA << endl;
14     }else{
15         cout << "Equipo B." << endl;
16         cout << puntajeB << endl;
17     }
18
19     return 0;
20 }
21
22 int etapa1(int *s1, int *s2){
23     cout << "---90 minutos---" << endl;
24     int golesA = 0;
25     cin >> golesA;
26     *s1 += golesA;
27
28     int golesB = 0;
29     cin >> golesB;
30     *s2 += golesB;
31
32     if(*s1 > *s2)
33         return 1;
34     else if(*s1 < *s2)
35         return 2;
36     else
37         etapa2(&s1, &s2);
38 }

```

Enviar
dirección de P

Indirección
simple

Desreferencia
simple

Desreferencia
simple

Enviar
dirección de S

```

40 int etapa2(int **d1, int **d2){
41     cout << "---Tiempo extra---" << endl;
42     int golesA = 0;
43     cin >> golesA;
44     **d1 += golesA;
45
46     int golesB = 0;
47     cin >> golesB;
48     **d2 += golesB;
49
50     if(**d1 > **d2)
51         return 1;
52     else if(**d1 < **d2)
53         return 2;
54     else
55         etapa3(&d1, &d2);
56
57 int etapa3(int ***t1, int ***t2){
58     cout << "---Tanda de penales---" << endl;
59     int golesA = 0;
60     cin >> golesA;
61     ***t1 += golesA;
62
63     int golesB = 0;
64     cin >> golesB;
65     ***t2 += golesB;
66
67     if(***t1 > ***t2)
68         return 1;
69     else
70         return 2;
71
72 }

```

Indirección
doble

Desreferencia
doble

Desreferencia
doble

Enviar
dirección de D

Indirección
triple

Desreferencia
triple

Desreferencia
triple

¿Dudas?

