Curso C

Punteros

punteros

- Un puntero es una variable que contiene la *direccion* de memoria de un dato o de otra variable que contiene el dato.
  - Declaracion:
    - Tipo *variable_puntero;
    - El * significa "*puntero a*"
  - Ejemplo:

```
int a=0;     //a es una variable entera (int)
int *pint;   //pint es un puntero a un entero
pint=&a;     //pint es igual a la direccion de a; pint
               apunta al entero "a"
```

Tanto a, como *pint son enteros. Luego….

Punteros – estructura memoria

- Punteros pueden aparecer donde aparecen las variables

```
#include <stdio.h>

main()
{
  int a=0;
  a=10;
  a=a-3;
  printf("%d",a);
}
```

```
#include <stdio.h>

main()
{
  int a=0,*pint=&a;
 *pint=10
 *pint=*pint-3;
  printf("%d",*pint);
}
```

- Explicar almacenamiento de programas en memoria.
  - Ojo, un puntero también tiene una dirección de memoria

punteros

- ¿Cómo sabe C cuantos bytes tiene que asignar a una variable desde una dirección?

```
#include <stdio.h>

main()
{
 int a=10,*q=NULL;
 double b=0.0;
 q=&a;
 b=*q;
 printf("En dir. %X esta dato %g\n",q,b);
}
```

- C toma como referencia el tipo del objeto para el que fue declarado el puntero (int) y asigna el numero de bytes correspondientes a ese tipo (4 bytes)
- Entonces …..

punteros

```
#include <stdio.h>

main()
{
  double a=10.33, b=0;
  int *p=NULL;
  p=&a;
  b=*q;
  printf("b=%g",b);
}
```

- Compilador avisa asignación de tipos incompatibles
- Resultado impredecible, C toma como referencia tipo (int) del puntero, asigna 4 bytes y no 8.
- Además, los 4 bytes se asignan como entero, no en formato IEEE 754 para coma flotante

Punteros: Operaciones

- Operador asignación ‘=‘ -> ambos punteros apuntarían a la misma dirección
- Operaciones aritméticas:
  - Podemos sumar o restar un numero entero
    - Implica que se desplaza tantas unidades como el tamaño del tipo del objeto al que apunta
  - NO PERMITIDAS:
    - Sumar, multiplicar, dividir, rotar ni sumarles un numero real.
  - PRIORIDADES:
    - El operador * y & tienen prioridad sobre los operadores aritméticos.

Punteros: Operaciones

- Comparación de punteros
  - Realmente estamos comparando enteros, pues el contenido de un puntero es una dirección (entera)
  - Comparar con NULL (constante indica puntero vacío) definida como ((void*)0)
- Punteros genéricos:
  - Un puntero a cualquier tipo puede convertirse al tipo void *
  - C permite conversion implicita (sin indicarlo nosotros) de un puntero generico (void*) a cualquier tipo;
  - C++ no lo permite, así que…. Mejor usar el casting

```
void *p;
int *a;
*a=10;
p=(void*) a;
```

Punteros: Operaciones

- Punteros constantes:
  - Declaración precedida de const hace que el objeto apuntado sea constante, pero no el puntero

```
int a=10, b=20;
const int *p=&a;

*p=15;  //ERROR, el objeto apuntado por p es constante
p=&b;   //CORRECTO, p variable y pasa a apuntar a otro objeto
```

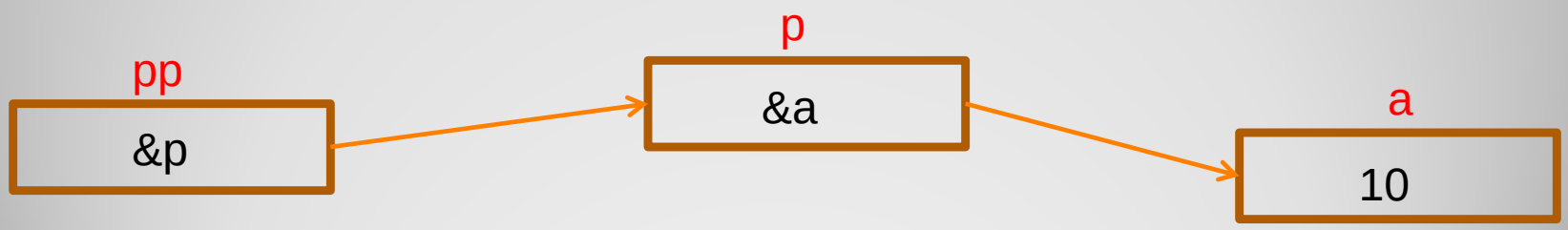
  - Si queremos puntero constante tendremos que hacer:

```
int a=10, b=20;
int * const p=&a;   //objeto variable y p constante

*p=15;  //CORRECTO, el objeto apuntado por p es variable
p=&b;   //ERROR, p es constante
```

Punteros: paso de parametros (POR REFERENCIA)

- Recordar paso parámetros a funciones (valor y referencia)
  - Nota: el puntero se pasa por valor, pero el objeto al que apunta se pasa por referencia
- Punteros a punteros:
  - Tipo **varpp;

```
int a, *p, **pp;

a=10; //Dato
p=&a; //puntero que apunta al dato
pp=&p; // puntero que apunta al puntero que apunta al dato
```



Punteros: punteros a funciones

- El nombre de una función representa la dirección donde se localiza dicha función.
  - Tipo (*p_identificador)();
  - Tipo es el tipo de dato del valor devuelto por la función
  - P_identificador es el nombre de la variable puntero

```
double (*pfn)();
double cuadrado(double);
double pot(double, double);

pfn=cuadrado;
pfn=pot;
```

Ambas son correctas, en C, la ausencia de parámetros indica cualquier numero y tipo de argumentos. En C++ no
- si queremos indicar no tiene parámetros poner void

Para invocar a la función: (*pfn)(argumentos);

Punteros y Matrices

- En C, cualquier operación que se pueda realizar con indexación de matrices, se puede hacer con aritmética de punteros.

```
main()
{
 int lista[]={24,30,15,45,34};
 int ind;
 for(ind=0;ind<5;ind++)
    printf("%d ",lista[ind]);
}
```

```
main()
{
 int lista[]={24,30,15,45,34};
 int ind;
 int *plista=&lista[0];
  for(ind=0;ind<5;ind++)
     printf("%d ",*(plista+ind));
}
```

lista[ind], *(lista+ind), plista[ind], *(plista+ind) SON EQUIVALENTES

```
main()
{
 int lista[]={24,30,15,45,34};
 int ind;
 int *plista=&lista;
  for(ind=0;ind<5;ind++)
     printf("%d ",*plista++);
}
```

Punteros y matrices

- Entonces ¿diferencia entre puntero e identificador de una matriz?
  - El identificador de una matriz en CONSTANTE y un puntero es variable

```
for(ind=0;ind<5;ind++)
    printf("%d ",*lista++);
```

ERROR, lista es puntero CONSTANTE

- Sin embargo, un parámetro de una función que sea una matriz es variable!!!!

```
void VisualizaMatriz(int lista[], int n)
{
   int ind;
   for(ind=0;ind<n;ind++)
      printf("%d ",*lista++);
}
```

CORRECTO !!!!

- OJO, sizeof aplicado a una matriz, devuelve tamaño en bytes, mientras que de un puntero devuelve el tamaño del puntero

```
int lista[]={24,30,15,45,34};
int *plista=&lista;
printf("%d %d",sizeof(lista), sizeof(plista));
```

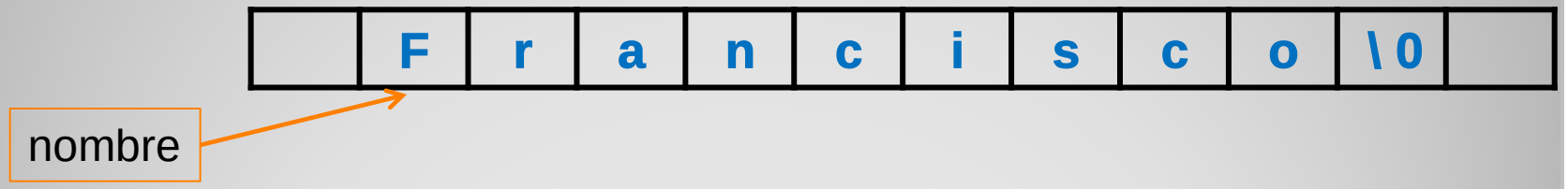
Imprimiría: 20 4

Ejercicio Copiar Matriz

Punteros y cadenas de caracteres

- Una cadena de caracteres es como un array o vector unidimensional
- ¿Cómo se identifica el principio y el final de una cadena de caracteres?
  - El nombre de la variable indica la dirección de memoria del principio de la cadena de caracteres.
  - El final se indica con el carácter '\0'

```
char *nombre="Francisco";
printf("%s",nombre);
```



- OJO, la variable nombre contiene la DIRECCION de memoria donde la cadena está almacenada

```
char *nombre="Francisco";
printf("%s",nombre);
nombre="Carmen";
```

ERROR, las constantes de caracteres no se pueden modificar

Ejercicios cadenas