Punteros

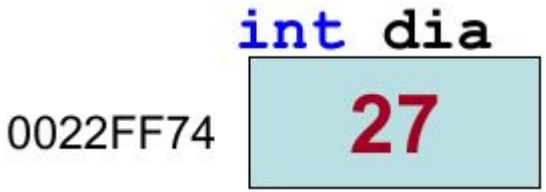
Prof. Dr. Hans H. Ccacyahuillca Bejar

Punteros

- **Introducción:**
  - Una variable es un espacio de la memoria principal, reservado para almacenar datos.
  - Variables poseen:
- **Nombre:**
  - Identificador para acceder al contenido
- **Tipo:**
  - Determina la capacidad de almacenamiento
  - Ex. int, char, float, ...
- **Dirección:**
  - Posición en la memoria principal

Punteros

- Ejemplo:
  - Nombre: `dia`
  - Tipo: `int`
  - Dirección: `0022FF74 (hexadecimal) o`
    `2293620 (decimal) o`
    `&dia (representación simbólica)`
  - Contenido: `27`



Punteros

- **Definición:**
  - Un puntero es una variable que almacena una dirección de memoria, como por ejemplo otra dirección de otra variable.
- **Declaración (sintaxis):**

```
tipo *nombre_puntero
```

- **Ejemplos:**

```
int *p;     /*declara puntero para un int*/
char *tmp; /*declara puntero para un char*/
float *punto /*declara puntero para un float*/
```

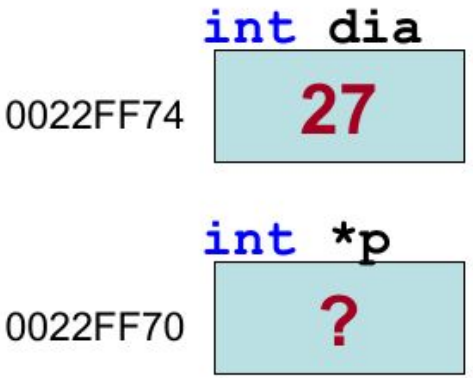
Punteros

Ejemplo:

```
#include <stdio.h>

int main(){
   int dia = 27;
   int *p;

   p = &dia;
   return 0;
}
```



- Las variables son declaradas
- Un puntero como toda variable también posee una dirección de memoria (&p=0022FF70

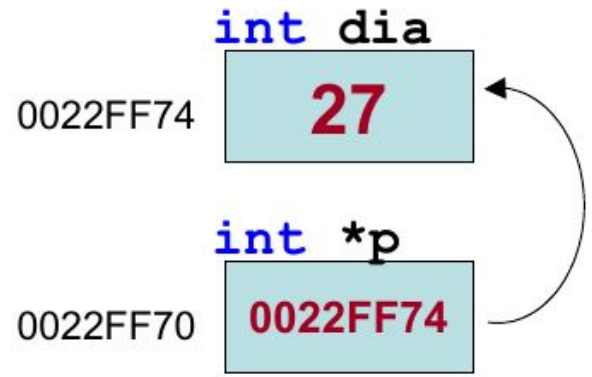
Punteros

Ejemplo:

```c
#include <stdio.h>

int main(){
  int dia = 27;
  int *p;

  p = &dia;
  return 0;
}
```



- La dirección de dia es atribuido para el puntero p.
- Decimos que p apunta para la variable dia (gráficamente representado por una flecha).

Punteros

- **Por qué usar punteros?**
  - En los ejemplos, el acceso al contenido de las **variables** se da a través del **nombre**.
  - Los punteros nos proporcionan un **nuevo modo** de acceso, que explora la **dirección** de las variables.
  - Para eso se usa el **operador indirecto** (*), que nos permite leer y alterar el contenido de las variables **apuntadas** por el puntero.

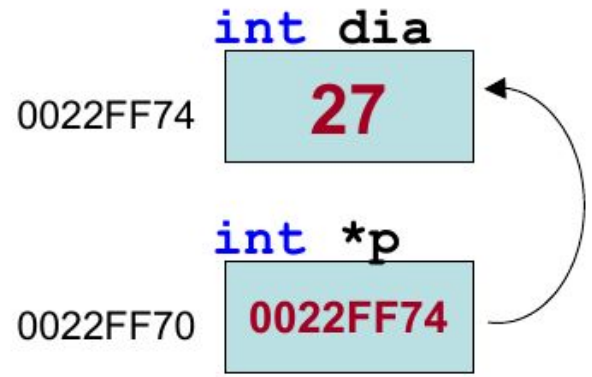
Punteros

Ejemplo:

```c
#include <stdio.h>

int main(){
  int dia = 27;
  int *p;

  p = &dia;
  return 0;
}
```



- La dirección de dia es atribuido para el puntero p.
- Decimos que p apunta para la variable dia (gráficamente representado por una flecha).

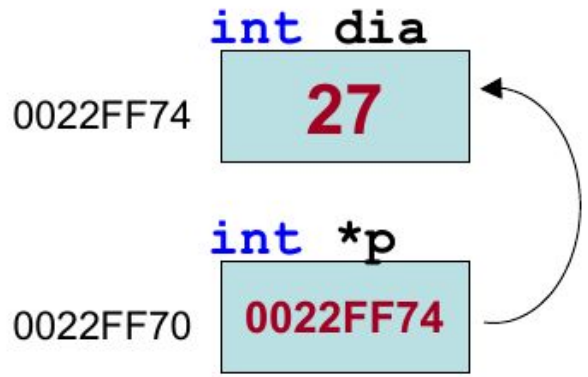
Punteros

Ejemplo:

```c
#include <stdio.h>

int main(){
  int dia = 27;
  int *p;

  p = &dia;
  return 0;
}
```



- El código *p es el contenido de la variable apuntada para p, osea el contenido de dia, que recibe el valor de 10.

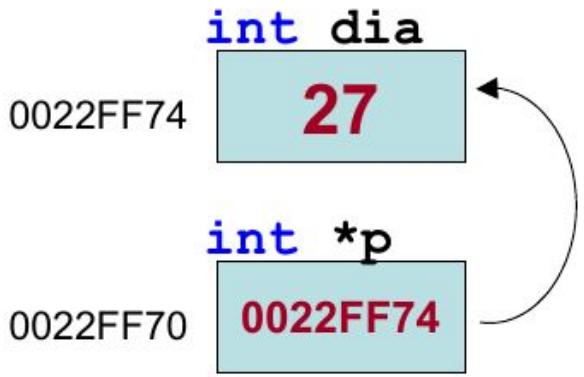
Punteros

Ejemplo:

```
#include <stdio.h>

int main(){
  int dia = 27;
  int *p;

  p = &dia;
  return 0;
}
```



- La declaración int *p; indica que la variable p es un puntero para un entero y que *p es de tipo int.

Punteros

- Por qué usar punteros?
  - Variables simples y estructuradas son pasadas como un valor en funciones. Osea, es generada una copia de la variable y las alteraciones en la función no producen ningún efecto externo.
  - Con el uso de punteros es posible realizar el paso de los valores por referencia.

Punteros

- **Ejemplos:** Función que lee un valor entero.

```c
#include <stdio.h>

void LeeEntero(int a){
     printf("Digite a: ");
     scanf("%d",&a);
}
int main(){
     int a=0;
     LeeEntero(a);
     printf("a: %d\n",a);
     return 0;
}
```

```c
#include <stdio.h>
int LeeEntero(int a){
     printf("Digite a: ");
     scanf("%d",&a);
     return a;
}
int main(){
     int a=0;
     a = LeeEntero(a);
     printf("a: %d\n",a);
     return 0;
}
```

- **Una posible solución es presentada a la derecha. Sin embargo, las funciones solo pueden devolver un único valor y en casos donde sea necesario modificar más de una variable el código no se aplica.**

Punteros

- **Ejemplos:** Función que lee un valor entero.

```c
#include <stdio.h>

void LeeEntero(int a){
     printf("Digite a: ");
     scanf("%d",&a);
}
int main(){
     int a=0;
     LeeEntero(a);
     printf("a: %d\n",a);
     return 0;
}
```

```c
#include <stdio.h>
void LeeEntero(int *p){
     int a;
     printf("Digite a: ");
     scanf("%d",&a);
     *p = a;
}
int main(){
     int a=0;
     LeeEntero(&a);
     printf("a: %d\n",a);
     return 0;
}
```

- Una solución es usando punteros. el puntero p es inicializado con la dirección de la variable a de la función principal. Luego, *p es el propio contenido de a de la función principal.
- Decimos que a es pasado por referencia.

Punteros

- **Ejemplo:** Función que cambia los valores de dos variables.

```
#include <stdio.h>
imt cambiar(int *a, int *b){
     int tmp;
     tmp = *a;
     *a = *b;
     *b = tmp;
}
int main(){
     int a=25,b=12;
     cambiar(&a,&b);
     printf("a: %d b: %d\n",a,b);
     return 0;
}
```

- La función cambiar es importante en algoritmo como Bubble sort. La función recibe las direcciones de dos variables y usa una variable temporal. La salida del programa es "a: 12, b: 25"

Ejercicio 1

En la función cambiar: Verificar el cambio de valores de las variables puede ser realizado mediante un macro de preprocesamiento:

```
#define cambiar(X, Y) {int t = X; X = Y; Y = t;}

. . .

cambiar(i, j);
```

Ejercicio 2

Por qué el siguiente codigo esta mal?

```
void cambiar (int *i, int *j) {
    int *temp;
    *temp = *i; *i = *j; *j = *temp;
}
```

Ejercicio 3

Un puntero puede ser usado para decirle a una función donde debe depositar el resultado de sus cálculos. Escribir una función `hm` que convierta minutos en hora-y-minutos. La función recibe un entero mnts y las direcciones de sus variables enteras, por ejemplo, `h` y `m`, asigna los valores a esas variables de modo que `m` sea menor que 60 y que `60*h + m` sea igual a mnts. Escriba también la función main que use la función `hm`.

Ejercicio 4

Escriba una función `mm` que reciba un vector entero `v[0..n-1]` y las direcciones de dos variables enteras, digamos `min` y `max`, y deposite en esas variables el valor del elemento mínimo y el valor del elemento máximo del vector. Escribir una función main que use la función `mm`.