

# Apuntadores

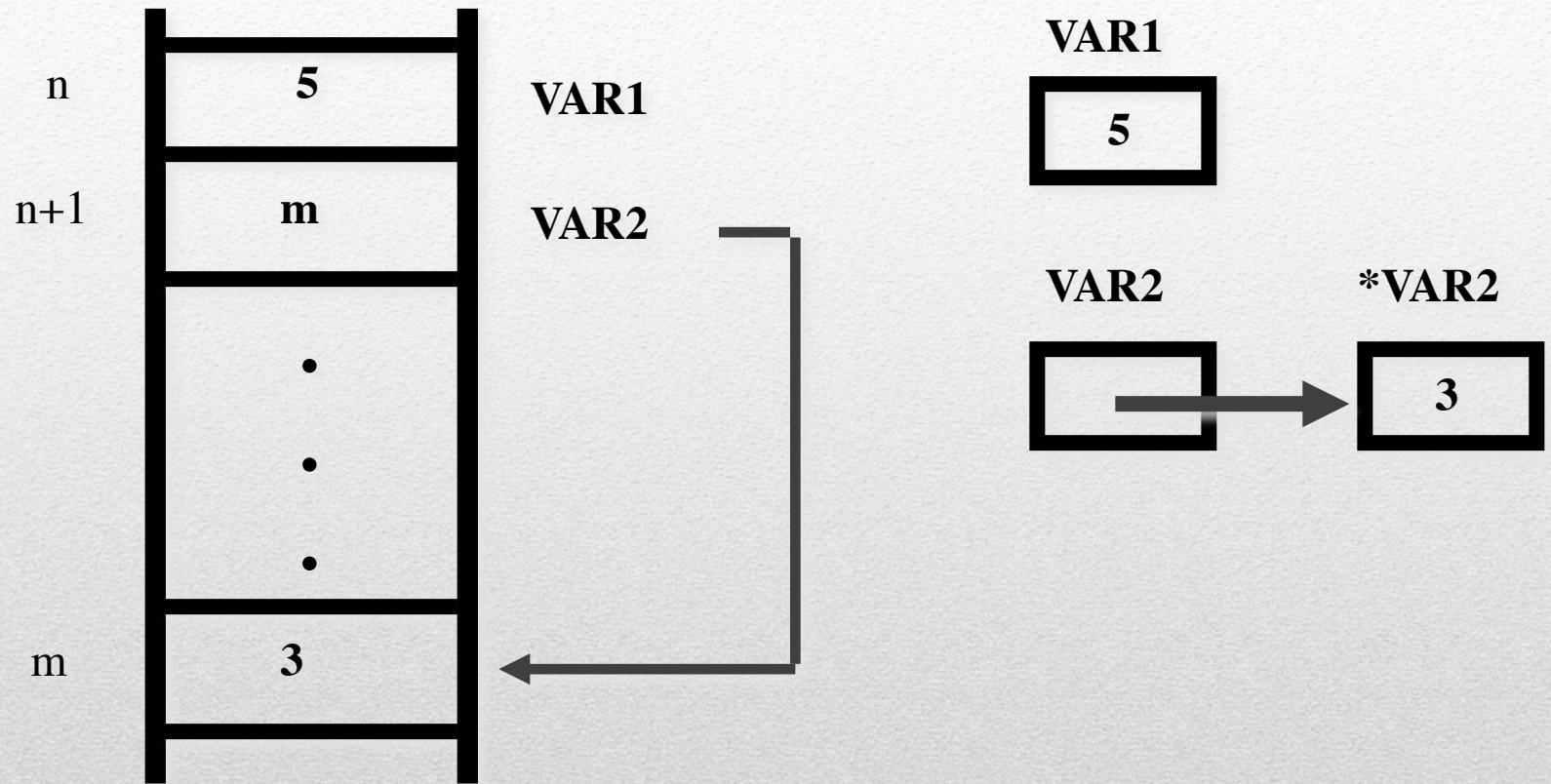
---

Ing. Luis Humberto González

- ✓ Un apuntador guarda una dirección de memoria donde cierto elemento en cuestión estará almacenado.
- ✓ Un apuntador es una referencia indirecta a un elemento.

# Apuntadores

---



# MEMORIA

---

- Declaración en C++:

tipo \*nombre\_apuntador;

- Ejemplos:

int \*p;

char c, \*s;

# Apun**t**adores

---

- Operador de indirección:

\*var\_apuntador

Hace referencia al espacio apuntado por la variable apuntador.... “*lo apuntado por...*”

- Operador de dirección:

&variable

Genera la dirección de la variable.

La variable puede ser de cualquier tipo.

# Operaciones...

---

- Modularidad : Paso de Parámetros por Referencia
- Manejo de Memoria Dinámica

# Aplicaciones...

---

# Modularidad... Ejemplo

Módulo para intercambiar 2 valores de variables:



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

*intercambia (z, w);*

# Ejemplo

---

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

*intercambia (&z, &w);*

**Ejemplo en C :**

---

```
void intercambia (int &a, int &b)
{ int temp;
  temp = a;
  a = b;
  b = temp; }
```

*intercambia (z, w);*

**Ejemplo en C++**

---

- Muchos lenguajes de programación tienen la capacidad de manejar dos tipos de almacenamiento de datos:
  - Estático (o Automático)
  - Dinámico

## **Tipos de memoria**

---

- Es la que se ha utilizado “tradicionalmente”.
- El espacio de memoria que se le asocia a una variable o a un objeto es **fijo** y permanece durante todo el tiempo de “vida” de la variable u objeto.

## Memoria Estática...

---

- Es la memoria que se define explícitamente al declarar una variable (global o local).
- Se llaman **Automáticas** porque son creadas automáticamente por el compilador al encontrar su declaración.

## **Memoria Estática...**

---

- El espacio de memoria en este tipo de almacenamiento se “crea” y se “destruye” según indicaciones del programador durante la ejecución del programa.
- Las variables y los objetos creados dinámicamente se localizan en el Heap, que es un área de memoria que administra el Sistema Operativo.
- Favorece el uso eficiente de la memoria en la ejecución de un programa.

## **Memoria Dinámica...**

---

- El espacio de almacenamiento dinámico se crea y se destruye usando los operadores:

new

delete

# Memoria Dinámica...

---

# Operadores new y delete...

- El operador new :  
apuntador = new tipo\_de\_dato;

Regresa la dirección del espacio de memoria que acaba de crear. Si no hubiera espacio, regresa el valor de cero (NULL). El apuntador toma el valor de la dirección y apunta a ese nuevo espacio.

- El operador delete:  
delete apuntador;

Libera el espacio apuntado que ya no se necesita, dejando al apuntador con un valor indefinido.

---

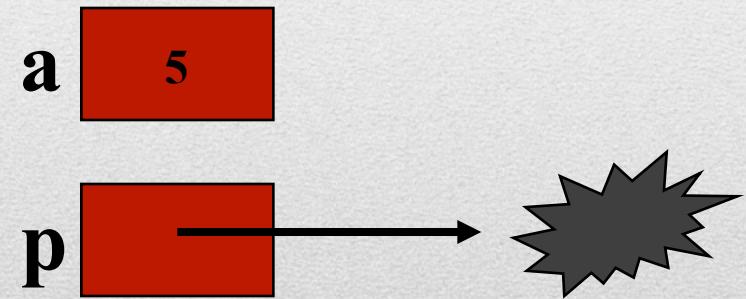
```
int a = 5, *p;  
p = new int;  
*p = a;  
a = *p + 2;  
cout << a << " " << *p;  
delete p;
```

*¿Qué se despliega en pantalla?*

# Ejemplo

---

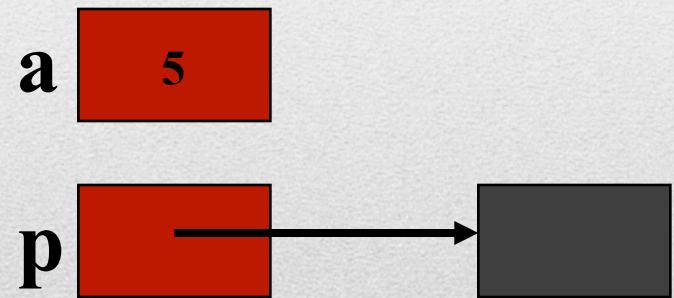
```
int a = 5, *p;
```



# Ejemplo

---

```
int a = 5, *p;  
p = new int;
```



# Ejemplo

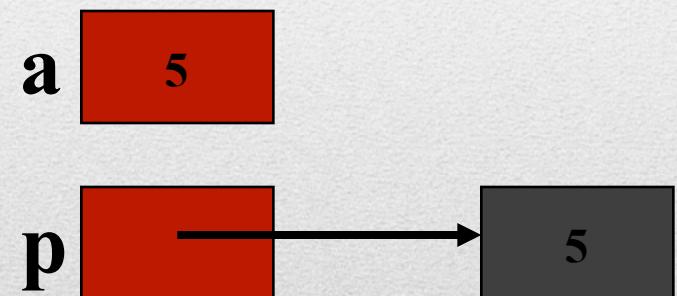
---

# Ejemplo

```
int a = 5, *p;
```

```
p = new int;
```

```
*p = a;
```



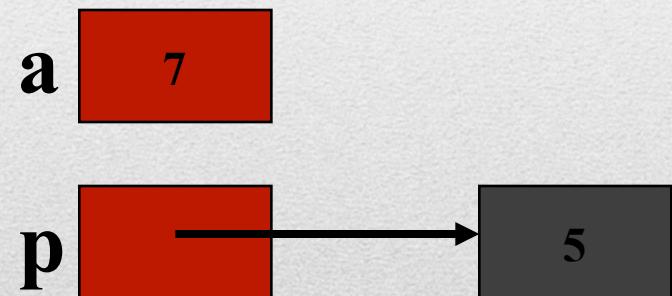
# Ejemplo

```
int a = 5, *p;
```

```
p = new int;
```

```
*p = a;
```

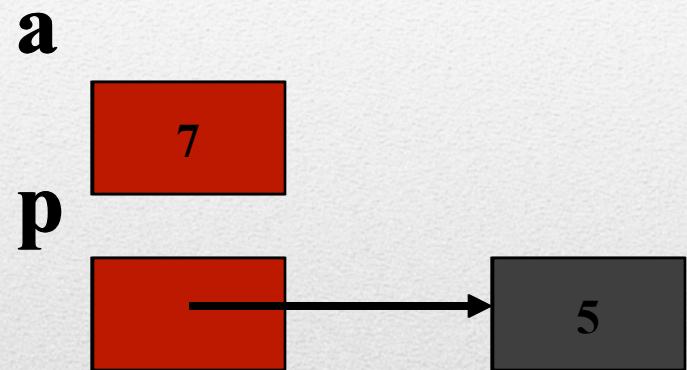
```
a = *p + 2;
```



```
int a = 5, *p;  
p = new int;  
*p = a;  
a = *p + 2;  
cout << a << " " << *p;
```

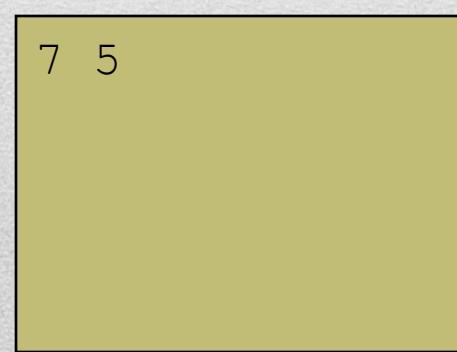
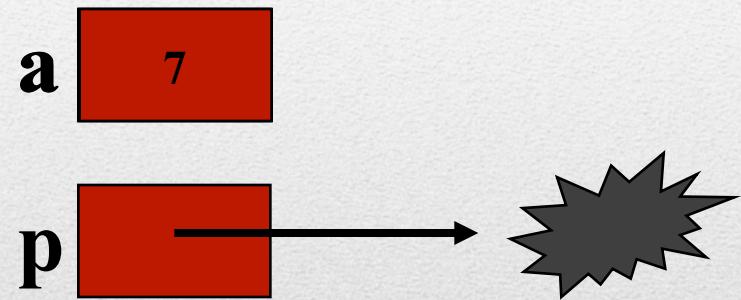
# Ejemplo

---



```
7 5
```

```
int a = 5, *p;  
p = new int;  
*p = a;  
a = *p + 2;  
cout << a << " " << *p;  
delete p;
```



# Ejemplo

---