Universidad Andrés Bello

Lenguajes de programación - Clase 4
Punteros – Parte 2

Definición Puntero

- Un puntero es una variable que contiene **la dirección de memoria** de un dato u otra variable.
- El puntero apunta al espacio físico donde está el dato o la variable.
- Los punteros pueden apuntar a cualquier tipo, como por ejemplo, una estructura o una función.
- Los punteros se utilizan principalmente para referenciar y manipular estructuras de datos, bloques de memoria asignados dinámicamente y proveer el paso de argumentos por referencia.
- Declaración:

```
tipo *nombrePuntero;
```

¿Por qué son útiles? Ejemplo didáctico

Universidad Andrés Bello

- ¿Cómo podríamos construir una función swap(a, b) que intercambie los valores entre a y b?
- Una idea:

```
void swap(int x , int y){
  int temp;

  temp = x;
  x = y;
  y = temp;
}
```

- ¿Se obtiene el intercambio en los valores?

¿Por qué son útiles? Ejemplo didáctico

Universidad Andrés Bello

- El intercambio no se logra, ya que las funciones solamente reciben valores como argumentos, por lo tanto, no podemos alterar la variable propiamente tal.
- Debemos recurrir a los punteros. ¿Por qué?

```
void swap(int *px , int *py){
  int temp;

  temp = *px;
  *px = *py;
  *py = temp;
}
```

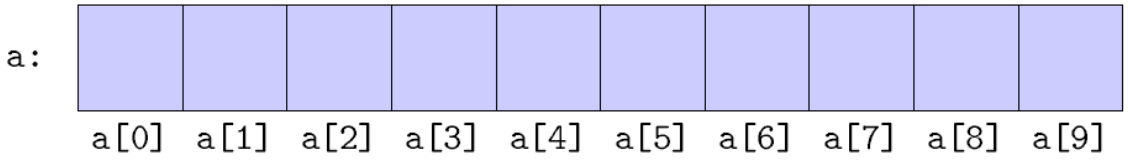
- Ahora debemos llamar a la función como swap(&a, &b) ¿Se obtiene el intercambio en los valores?

Punteros y Arreglos

- En C, hay una poderosa relación entre los punteros y los arreglos.
- Cualquier operación que se realiza con los índices de un arreglo, es posible realizarlo con los punteros.
- La declaración:

```
int a[10];
```

define un arreglo a de tamaño 10, el cual es, un bloque de 10 objetos consecutivos denominados a[0], a[1], ..., a[9]



Punteros y Arreglos

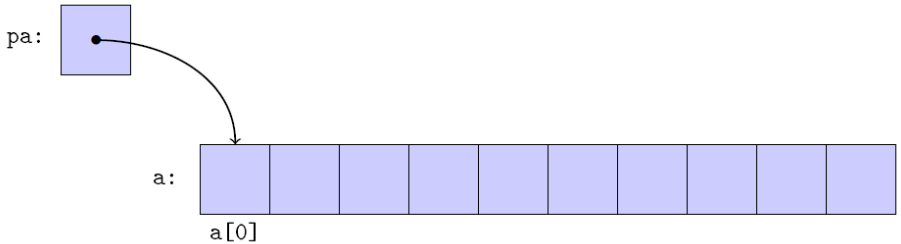
- La notación a[i] se reere al i-ésimo elemento del arreglo.
- Si pa es un puntero a un entero declarado como:

```
int *pa;
```

- Entonces la asignación:

```
pa = &a[0];
```

- permite que pa apunte al primer elemento de a, esto es, pa contiene la dirección de a[0]:



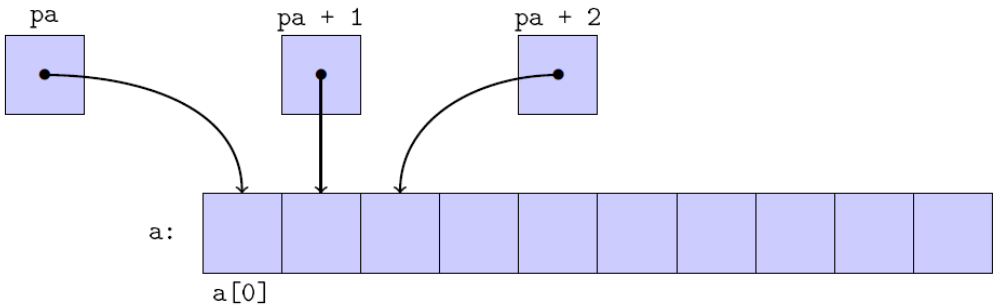
Punteros y Arreglos

- Luego la asignación

```
x = *pa;
```

copiará el contenido de a[0] en x.

- Si pa apunta a un elemento en particular de un arreglo, entonces por denición pa + 1 apunta al próximo elemento, luego pa + i apunta al i-ésimo elemento después de pa y pa-i apunta al i-ésimo elemento antes.



Punteros y Arreglos

- Dado que el nombre del arreglo es sinónimo de la ubicación del elemento inicial, la asignación pa = &a[0] también se puede escribir como:

```
pa = a;
```

- Una referencia a a[i] también se puede escribir como *(a + i), al igual que pa[i] es idéntico a *(pa + i).
- **Importante:** Un puntero es una variable, luego pa = a y pa++ es válido, pero un arreglo no es una variable, por lo tanto a = pa y a++ no son expresiones válidas.

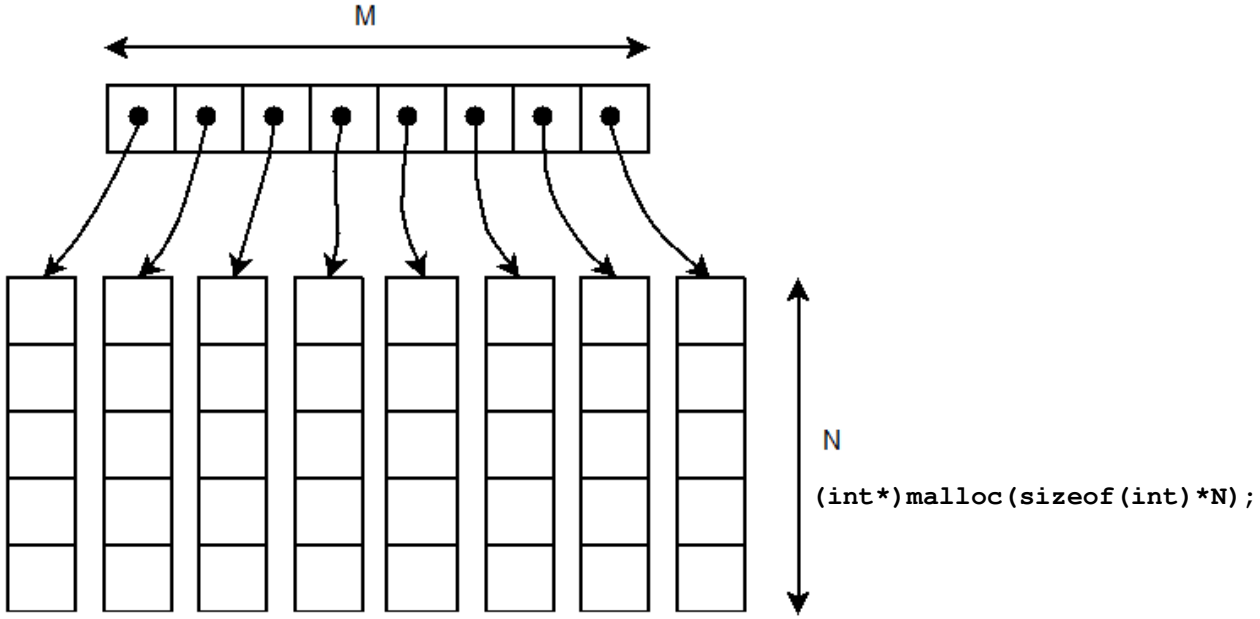
Arreglo de Punteros

Universidad Andrés Bello

- Dado que los punteros son variables, pueden ser almacenados en arreglos.
- Por ejemplo, podríamos ordenar un conjunto de líneas de texto en orden alfabético.
- ¿Cómo entonces almacenamos, en primer lugar las líneas de texto si son de distinto tamaño?
- Necesitamos una representación de datos que nos permita almacenar eficientemente cada línea de texto.
- **Solución:** arreglo de punteros.

Arreglo de Punteros

```
(int**)malloc(sizeof(int*)*M);
```




```
(int*)malloc(sizeof(int)*N);
```

Universidad Andrés Bello

No olvidar!

repl.it