

# Laboratorio de Programación

## Lenguaje C Básico: Arreglos, Punteros y Funciones

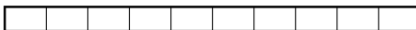
Luis Garreta  
luis.garreta@javerianacali.edu.co

Ingeniería de Sistemas y Computación  
Pontificia Universidad Javeriana – Cali

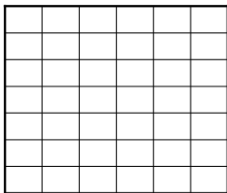
14 de febrero de 2017

# Arreglos

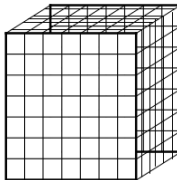
- Los arreglos son una colección de variables del mismo tipo.
- Los elementos individuales son identificados por un índice entero:
  - Índice comienza en cero y termina en n-1
  - Índice siempre es escrito dentro de corchetes `[]`.
  - Acceso a los valores a través del operador `[]`
- Pueden ser de distintas dimensiones:



**Unidimensional** `int a[20];`



**Bidimensional** `int a[6][7];`



**Tridimensional** `float a[7][7][4];`

# Ejemplos de Arreglos en C

```
1  #include <stdio.h>
2  int main () {
3      int i; float valor;
4      int  ai  [] = {2,4,6,8};
5      float af [4];
6
7      for (i=0; i < 4; i++)
8          printf ("El valor %d es %d\n", i, ai [i]);
9
10         printf ("Digite valor %d: ", i);
11         scanf ("%f", &valor);
12         af [i] = valor;
13     }
14     return 1;
15 }
```

# Introducción a Punteros (apuntadores)

- Puntero: es una variable:
  - Guarda la dirección de memoria de otra variable (específica)
  - Cuando se declara no guarda espacio para la variable.
- Operadores:
  - Referencia (\*): accede al contenido de la dirección de memoria guardado por el puntero.
  - Desreferencia (&): obtiene la dirección de memoria de una variable.

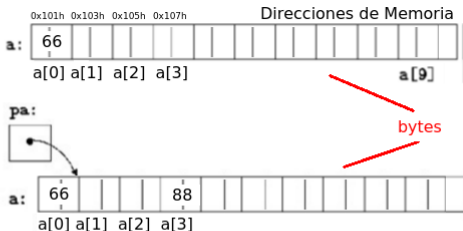
# Arreglos y Punteros

- `int a[10];`  
`a[0] = 66;`

- `int *pa;`

- `pa = a;`  
`pa[3] = 88;`

- `x = *pa /*copia el contenido de a[0] en x */`  
`printf ("%d". x); /* Imprime 66 */`



# Relación entre arreglos y punteros

- El nombre de un arreglo es un puntero al inicio del bloque de memoria del arreglo

```
1  int a1[] = {1, 2, 3, 4, 5};  
2  int* p1;  
3  p1 = a1; // equivalente a p1 = &a1[0];
```

- Aritmética de punteros: útil para recorrerlos

```
1  *p1 = 10; // equivalente a ai[0] = 10;  
2  p1++;  
3  *p1 = 12; // equivalente a ai[1] = 12;
```

# Ejemplos de Punteros como Arreglos

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Contruccion de un arreglo dinámico de n datos */
4  int main () {
5      int i, n, a1 [] = {2,4,6,8};
6      int *p1, *p2; // equivalente a int *p1
7
8      p1 = a1; // p1 apunta a la misma dir que a1
9      for (i=0; i<4; i++)
10         printf ("%d\n", p1 [i]);
11
12     scanf ("%d", &n); // Lee el número de valores
13
14     p2 = (int *)malloc (n * sizeof (int));
15     for (i=0; i < n; i++)
16         scanf ("%d", &p1 [i]);
17
18     return 0;
19 }
```

# String: cadena de caracteres

- No está soportado directamente por C.
- Características de los strings:
  - arreglo de caracteres
  - deben terminar en carácter nulo (**NULL**): *fin de cadena*.

```
1 char* frase = {'H', 'o', 'l', 'a', '\0'}; //  
    equivalente a:  
2 char frase[] = "Hola"; //compilador agrega caract. '\0'
```

- La biblioteca string.h provee funciones para manipulación de cadenas:

```
1 char *strcpy(char *dest, const char *src);  
2 int strcmp(const char *s1, const char *s2);  
3 int atoi(const char *nptr);
```



## Ejemplo de Funciones de Cadenas

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  int main() {
5      char cadena1 [] ="aprenderaprogramar.com"; char cadena2
        [sizeof(cadena1)];
6      char cadena3[] = " ok!";      char cadena4[50];
7      strcpy (cadena2, cadena1); //Copia cadena1 en cadena2
8      printf ("Cadena1 %s y Cadena2 %s \n", cadena1, cadena2);
9      printf ("Longitud cadena1 %d\n", strlen(cadena1));
10
11     strcpy (cadena4, cadena1) ;
12     printf ("Concatenacion %s\n", strcat(cadena4, cadena3));
13     printf ("Cadena4 vale %s \n", cadena4);
14
15     if (strcmp (cadena4, cadena2)==0)
16         puts ("Las cadenas 1 y 2 son iguales");
17     if (strcmp (cadena1, cadena3) > 0 )
18         puts ("La cadena 1 precede alfabeticamente a la
            cadena 3");
19     return 0;
20 }
```

# Otras Funciones de Cadenas

**strlen** - Finds out the length of a string  
**strlwr** - It converts a string to lowercase  
**strupr** - It converts a string to uppercase  
**strcat** - It appends one string at the end of another  
**strncat** - It appends first n characters of a string at the end of another.  
**strcpy** - Use it for Copying a string into another  
**strncpy** - It copies first n characters of one string into another  
**strcmp** - It compares two strings  
**strncmp** - It compares first n characters of two strings  
**strcmpi** - It compares two strings without regard to case ("i" denotes that this function ignores case)  
**stricmp** - It compares two strings without regard to case (identical to strcmpi)  
**strnicmp** - It compares first n characters of two strings, Its not case sensitive  
**strdup** - Used for Duplicating a string  
**strchr** - Finds out first occurrence of a given character in a string  
**strrchr** - Finds out last occurrence of a given character in a string  
**strstr** - Finds first occurrence of a given string in another string  
**strset** - It sets all characters of string to a given character  
**strnset** - It sets first n characters of a string to a given character  
**strrev** - It Reverses a string