Estructuras de Datos

Luis Garreta
luis.garreta@javerianacali.edu.co

Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

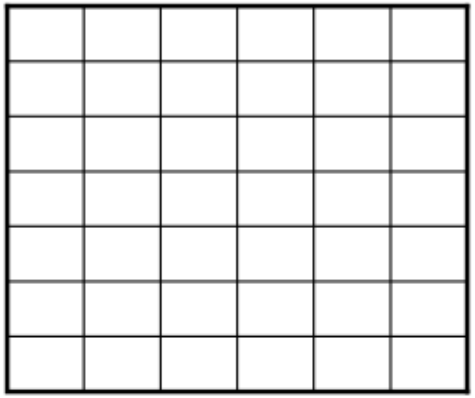
17 de agosto de 2017

Arreglos, Punteros y Cadenas

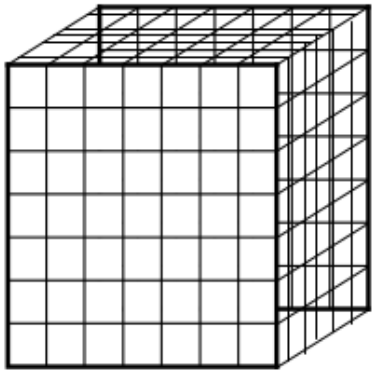
Arreglos

- Los arreglos son una colección de variables del mismo tipo.
- Los elementos individuales son identificados por un índice entero:
  - Indice comienza en cero y termina en n-1
  - Indice 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

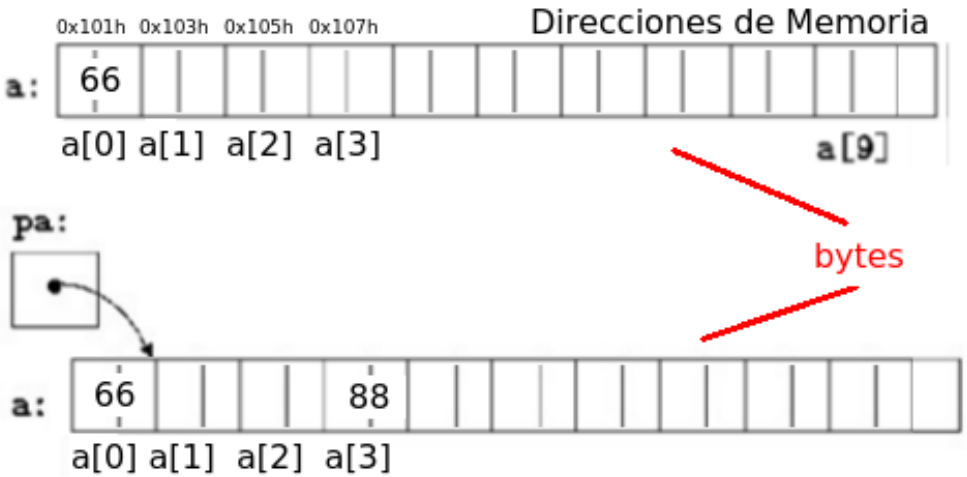
```c
#include <stdio.h>
int main () {
  int i; float valor;
  int   ai  [] = {2,4,6,8};
  float af [4];

  for (i=0; i < 4; i++)
    printf ("El valor %d es %d\n", i, ai [i]);

    printf ("Digite valor %d: ", i);
    scanf ("%f", &valor);
    af [i] = valor;
  }
  return 1;
}
```

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 pi = &ai[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

```c
#include <stdio.h>
#include <stdlib.h>
/* ContrucciOn de un arreglo dinAmico de n datos */
int ma1n () {
  int i, n, a1 [] = {2,4,6,8};
  int *p1, *p2; // equivalente a int *p1

  p1 = a1; // p1 apunta a la misma dir que a1
  for (i=0; i<4; i++)
    printf ("%d\n", p1 [i]);

  scanf ("%d", &n); // Lee el número de valores

  p2 = (int *)malloc (n * sizeof (int));
  for (i=0; i < n; i++)
    scanf ("%d", &p1 [i]);

  return 0;
}
```

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 [] ="lenguaje C++ ";
6      char cadena2 [sizeof(cadena1)];
7      char cadena3[] = " ok!";
8      char cadena4[50];
9      char *cadena5;
10
11     strcpy (cadena2, cadena1); //Copia cadena1 en cadena2
12     n = strlen(cadena1);
13
14     cadena5 = strcat(cadena1, cadena3)); // "
15
16     if (strcmp (cadena4, cadena2)==0)
17       puts ("Las cadenas 1 y 2 son iguales");
18     if (strcmp (cadena1, cadena3) < 0 )
19       puts ("Cadena 1 menor alfabeticamente que cadena 3");
20
21     return 0;
22 }
```

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