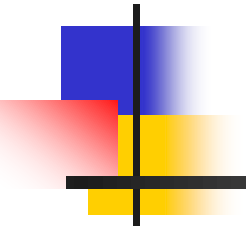


PROGRAMACIÓN ESTRUCTURADA

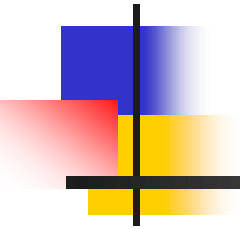
UNIDAD 4.- ARREGLOS

COMPETENCIA:

Desarrolla programas computacionales que requieren del manejo de conjuntos de datos, mediante el uso de la estructura arreglo.

SECUENCIA DE CONTENIDOS:

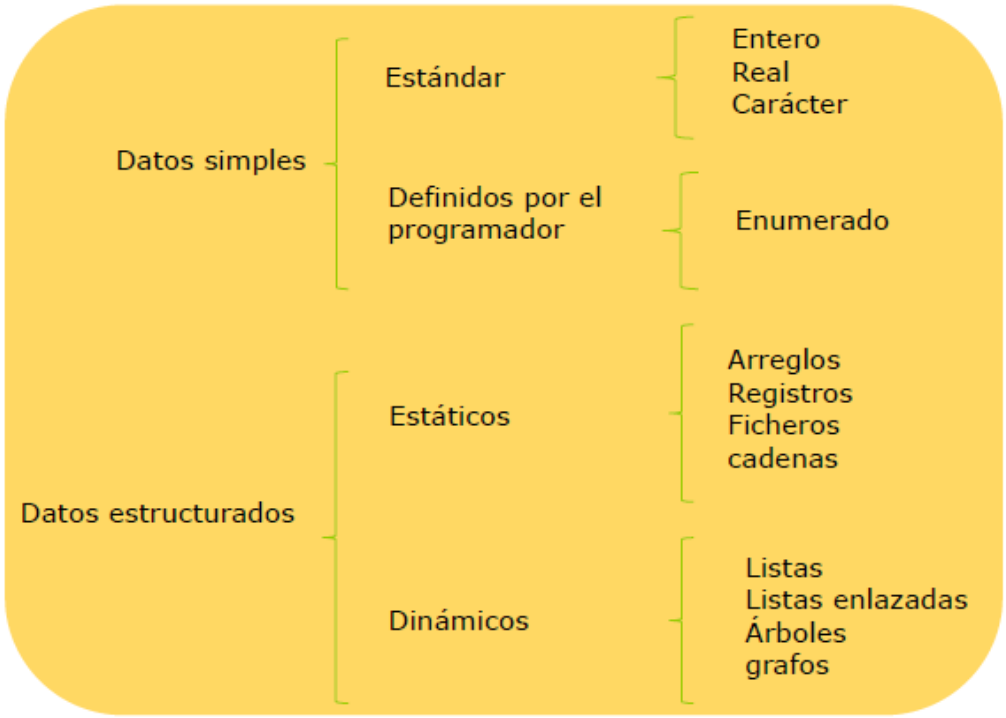
1. Introducción a las estructuras de datos.
2. La estructura arreglo.
3. Cadenas.



1. Introducción a las estructuras de datos

Estructuras de datos

- Una **estructura de datos** es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella.
- Tipos de datos más frecuentes:



Estructuras de datos

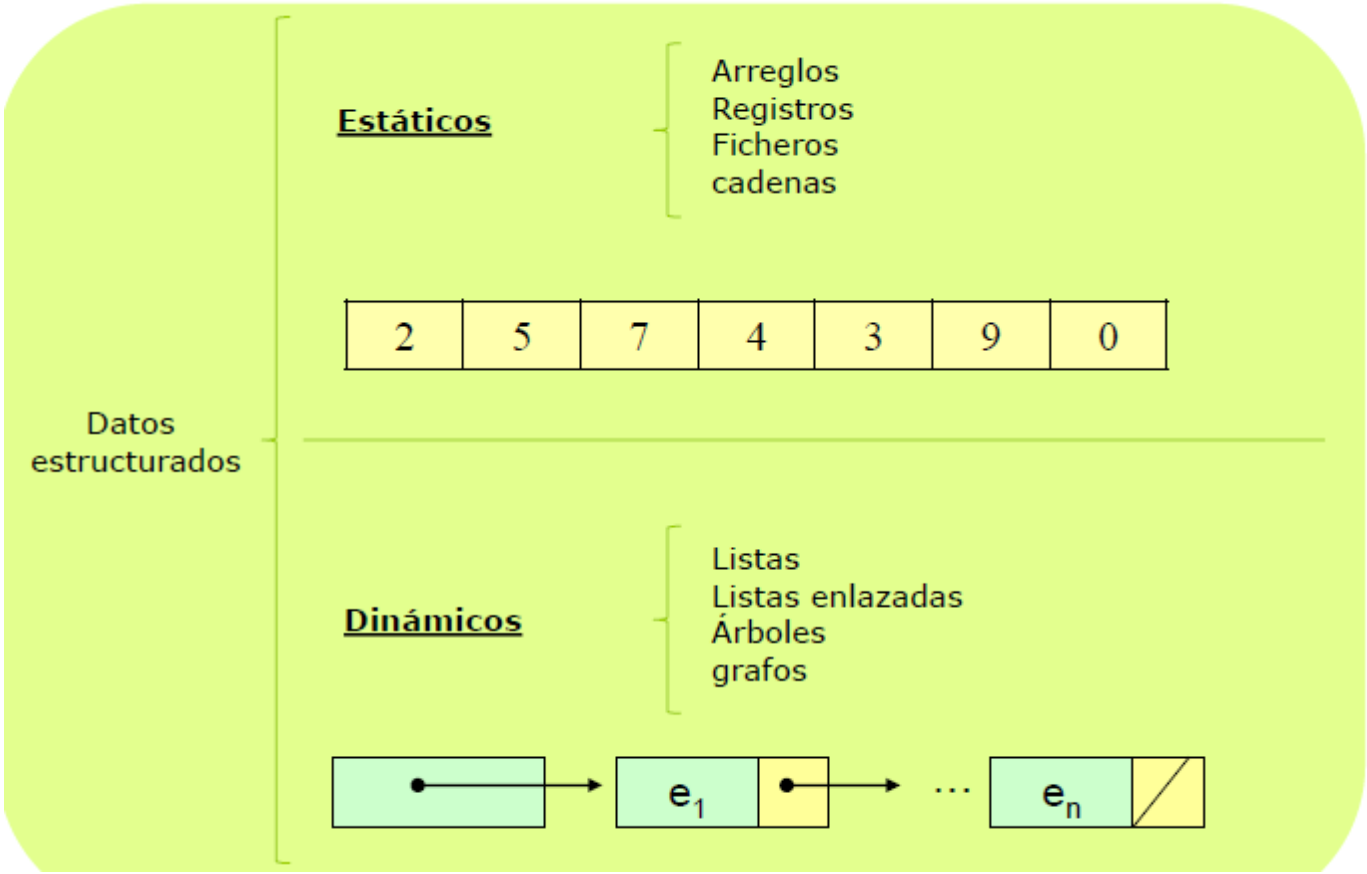
- Los tipos de datos simples o primitivos significan que no están compuestos de otras estructuras de datos. Los tipos de datos compuestos están construidos basados en tipos datos primitivos.

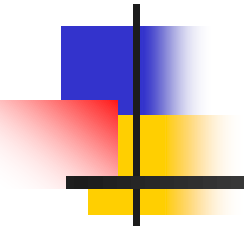
- Los tipos de datos simples pueden ser organizados en estructuras de datos **estáticas** y **dinámicas**.

  - **Estáticas**.- El tamaño ocupado de memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa.
  - **Dinámicas**.- No tienen limitaciones o restricciones en el tamaño de memoria ocupada.

Estructuras de datos

- Estructuras de datos estáticas y dinámicas:





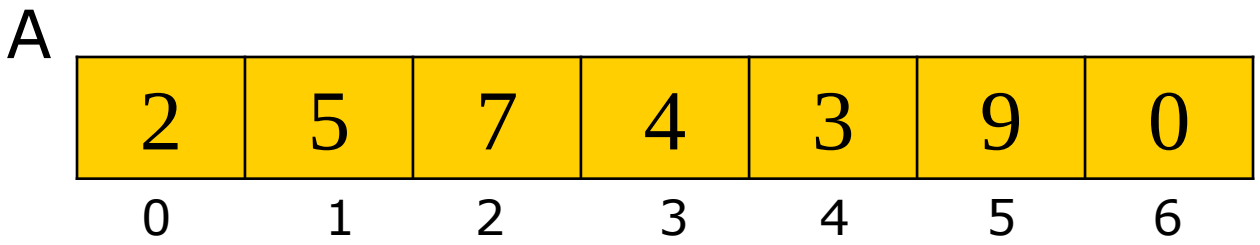
2. La estructura arreglo

Arreglos

- Un arreglo, es una secuencia de datos del mismo tipo. Los datos se llaman elementos del arreglo y se numeran consecutivamente 0, 1, 2, …etc. A la enumeración de los elementos, se le llama subíndices tal y como se toma en matemáticas: a_0, a_1, a_2…. Los subíndices proporcionan la posición del elemento dentro del arrglo, es decir, el acceso al elemento es directo.

- Ejemplo:

  Sea A el nombre del arreglo:



Entonces A[0], es el valor del elemento en la posición 0, A[1] es el valor del elemento en la posición 1, etc.

El lenguaje C inicia el subíndice en 0, por lo que si el array tiene elementos, los valores del array serían A[0], A[1], A[2],…,A[n-1].

Arreglos: declaración

- Un arreglo se declara de modo similar a otros tipos de datos, excepto que se debe indicar al compilador el tamaño o longitud del arreglo. Para indicar al compilador el tamaño o longitud del arreglo, seguido del nombre se coloca el tamaño encerrado entre corchetes. La sintaxis para declarar un arreglo de una dimensión determinada es:

  Sintaxis de un arreglo:

  tipo nombreArray[numeroElementos];

Ejemplos:

```
int edad[40];
float salario_empleados[200];
double temperaturas[50];
char letras[15];

#define N 20
float vector[N];
```

Inicialización de arreglos

- Se deben asignar valores a los elementos del array antes de utilizarlos.

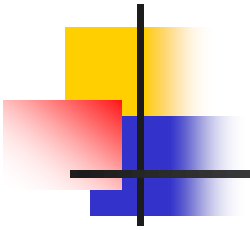
```
precios[0] = 10;
precios[1] = 20;
int numeros[6] = {10, 20, 30, 40, 50, 60};
int n[] = {3, 5, 8};
char c[] = { 'L', 'U', 'I', 'S'};

Inicializar arreglo de caracteres
    char s[] = "Al fin es viernes!!";
```

Ejemplo

- Imprimir la suma de 8 números definidos en un arreglo.

```
#define NUM 8
int main(){
        int nums[NUM];
        int i, total=0;
        for(i=0;i<NUM; i++){
         printf("Intdoduzca el numero: ");
         scanf("%d", &nums[i]);
}
printf("\nLista de numeros ");
for(i=0;i<NUM; i++){
        printf("%d", nums[i]);
        total+=nums[i];
}
 printf("\nLa suma de los elementos del arreglo
es:%d ",total);
return 0;
}
```



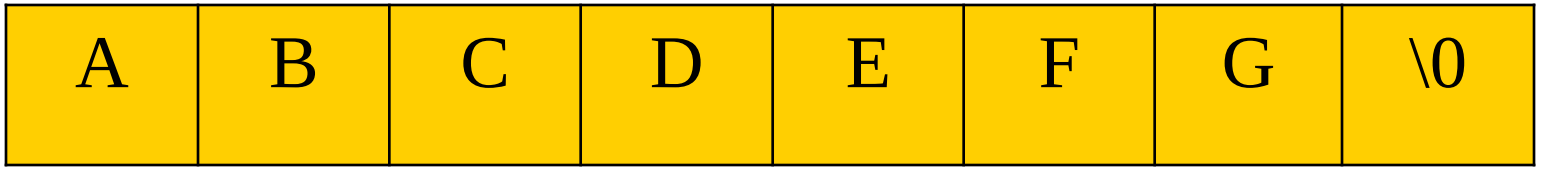
¡Importante!

- Al definir un arreglo global o estático y no se proporciona ningún valor de inicialización, el compilador inicializará el arreglo con un valor por defecto:
  - (cero para arreglo de enteros y flotantes y carácter nulo para arreglos de caracteres).

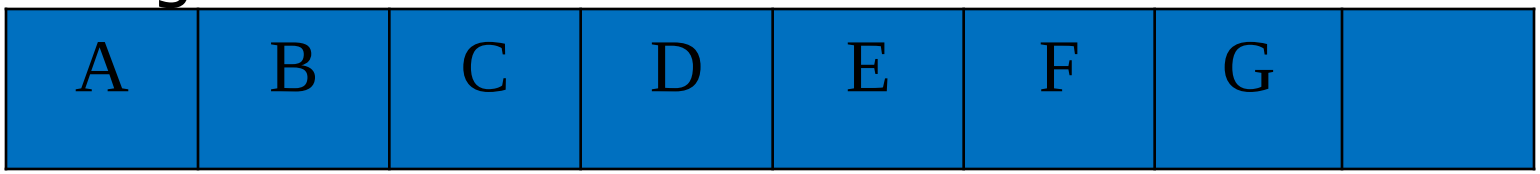
Cadenas y arreglo de caracteres

- Las cadenas se señalan incluyendo el carácter nulo (\0) al final de la cadena, cuyo valor en el código ASCII es 0.
- Ejemplo:
  - char cadena[7] = "ABCDEFG";
- El compilador inserta automáticamente un carácter nulo al final de la cadena, de modo que la secuencia real sería:

cadena

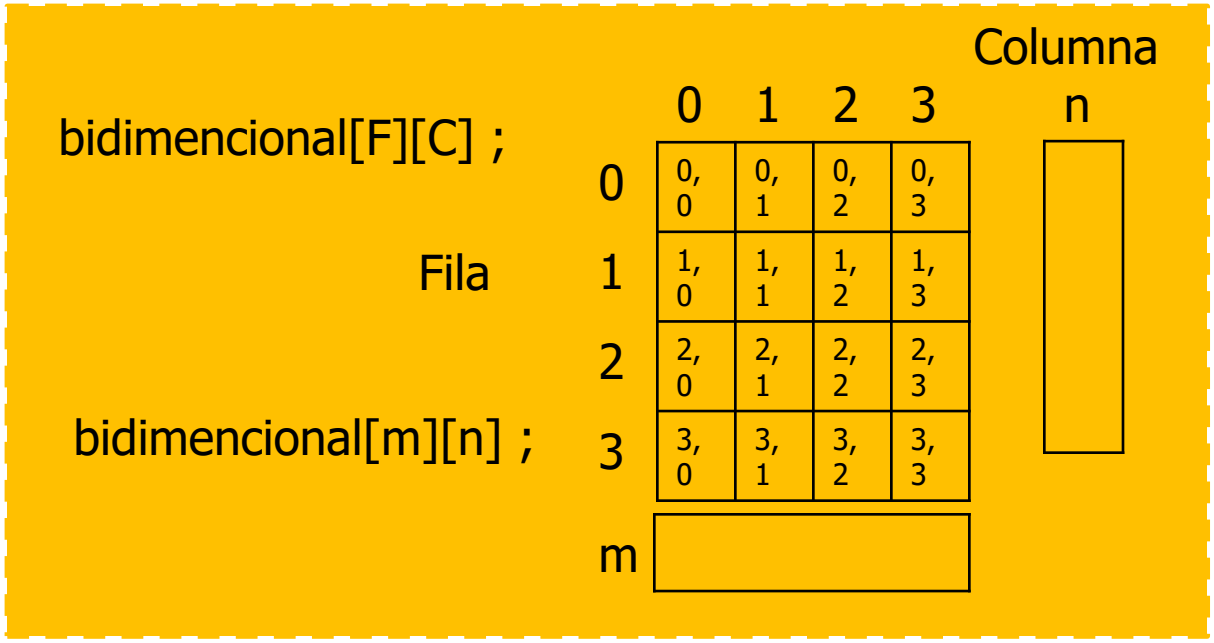


Arreglo de caracteres



Arreglos multidimensionales

- Los arreglos multidimensionales son aquellos que tienen más de una dimensión y, en consecuencia, más de un índice. Los arreglos más usuales son los de dos dimensiones, conocidos también como tablas o matrices. Sin embargo, es posible crear arreglos de tantas dimensiones como requieran sus aplicaciones, esto es, tres, cuatro o más dimensiones.



Declaración e inicialización

- Declaración:

  tipo_de_dato nombre_array [num_fil][num_col];

  ```
  char pantalla[25][80];
  int impuestos[6][8];
  int miArreglo[4][12];
  ```

- Inicialización:

  ```
  int tabla[2][3] = {23, 34, 12, 78, 14, 98};
  int tabla[2][3] = {  {23, 34, 12},
                       {78, 14, 98} };
  int tabla[2][3] = { {23, 34, 12}, {78, 14, 98} };
  ```

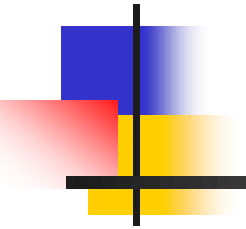
Acceso a los elementos mediante bucles

```
#include <stdio.h>
int main() {
      float matriz[4][4];
      int fila, col;
      for (fila = 0; fila < 4; fila++){
            for (col = 0; col < 4; col++){
                printf ("\n Introduzca un valor en la Posicion [%d][%d]:",fila,col) ;
                scanf ("%f",&matriz[fila][col]);
            }
      }
      for (fila = 0; fila < 4; fila++){
            for (col = 0; col < 1; col++){
                printf ("\n %.1f | %.1f | %.1f | %.1f\n",matriz[fila][col],
                              matriz[fila][col+1], matriz[fila][col+2],
      matriz[fila][col+3]) ;
            }
      }
       return 0;
}
```

Ejemplo

Determina si la matriz es simétrica

```
int simetrica(int a[][N],int n){
    int i,j;
    int es_simetrica;
    for (es_simetrica=1, i=0; i<n-1 && es_simetrica; i++)
            for (j=i+1; j<n && es_simetrica; j++)
                    if(a[i][j] != a[j][i])
                      es_simetrica=0;
  return es_simetrica;
}
```



3. Cadenas

Definición de cadena

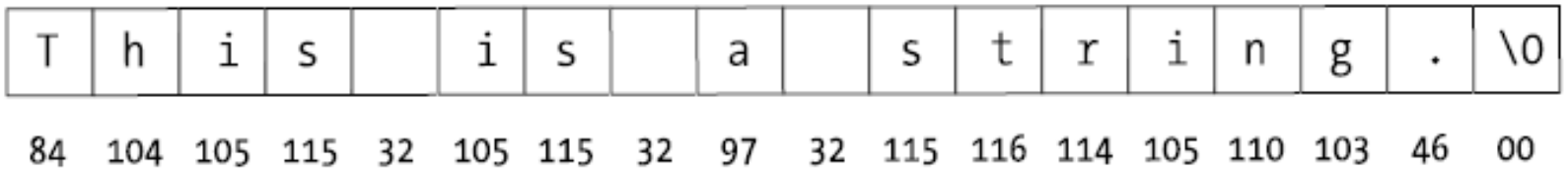
- Una cadena es una secuencia de caracteres, almacenados en posiciones de memoria contiguas, que termina con el carácter nulo. Es un tipo de dato compuesto, es un arreglo de caracteres (char), terminado por un carácter nulo (\0).

- Ejemplo: "ABC"
  - Cuando la cadena aparece dentro de un programa se verá como si se almacenaran cuatro elementos: 'A', 'B', 'C' y '\0'

Ejemplo

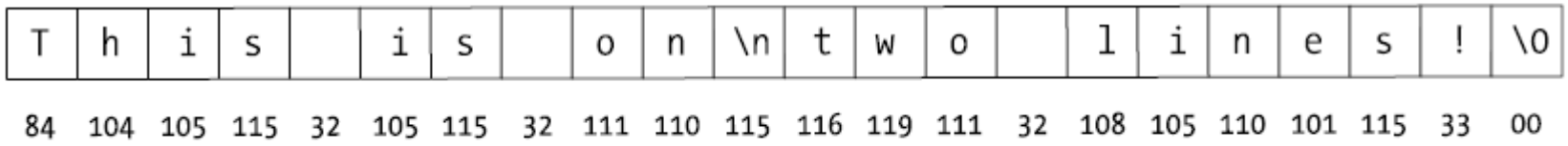
printf("This is a string.");

"This is a string."



printf("This is on\ntwo lines!\n");

"This is on\ntwo lines!"



Declaración de cadenas

Tipo_dato nombre_cadena[longitud];

- Ejemplos:

```
char texto[8];
char orden[40];
char texto[81] = "Esto es una cadena";
char nombre[50] = "Juan Zapata";
char cadena[] = "¿Cuanto es la longitud de esta cadena?";
```

- Una cadena no se puede inicializar fuera de la declaración.

Ejemplos

- Programa que visualiza los valores de una cadena:

```
int main(){
   char s[] = "ABCD";
   for (int i=0;i<5;i++)
      printf("s[%d] = %c\n", i, s[i]);
   return 0;
}
```

- Programa que lee e imprime una cadena:

```
int main(){
      char nombre[30];
      scanf("%s",nombre);
      printf("%s",nombre);
      return 0;
}
```

Ejemplos

- **Cuenta caracteres de una cadena**

```
int cuenta_caracteres(char carac[]) {
   int i=0;
   while (carac[i] != '\0')
        i++;
    return i;
}
```

- **Compara dos cadenas**

```
int compara_cadenas(char carac1[],
     char carac2[])
{
    int i = 0, dif;
    while (carac1[i] != '\0') {
        dif = carac1[i] - carac2[i];
        if (dif == 0)
            i++;
        else
        return (dif);
    }
    if (carac2[i] == '\0')
        return(0);
}
```

Utilizando la Librería string

- Para utilizar la librería String hay que incluir "string.h" en el programa:

 # include <string.h>

- Ejemplos de funciones que devuelven un "int":

  strcmp (comparar)

  strncmp (comparar "n" elementos)

  strlen (longitud)

Ejemplo: Comparando dos string

```
# define TAM 40
# include <stdio.h>
# include <string.h>
int main() {
      char s1[TAM], s2[TAM];
      int n;
      printf("Introducir dos cadenas: ");
      scanf("%s %s", s1, s2);
      n = strcmp(s1, s2);
      if ( n == 0 ) printf("\nHa introducido dos cadenas iguales");
      else if ( n<0 ) printf("Hay un caracter diferente en s1 y es menor que el de s2\n");
      else printf("\nHay un caracter diferente en s1 y es mayor que el de s2\n");
      return 0;
}
```

Ejemplo: Búsqueda en string

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str1[] = "El desarrollo de software es un proceso iterativo";
    char str2[] = "software";
    char str3[] = "proceso iterativo";
    if(strstr(str1, str2) == NULL)
        printf("\n\"%s\" No se encontro en el string1.", str2);
    else    printf("\n\n\"%s\" se encontro en el string1 \"%s\"",str2, str1);
    if(strstr(str1, str3) == NULL)
        printf("\n\"%s\" no se encontro en el string1.", str3);
    else    printf("\n\"%s\" se encontro en el string1 \"%s\"",str3, str1);
    return 0;
}
```

Funciones de la librería ctype.h

Table 6-1. *Character Classification Functions*

Function	Tests For
`islower()`	Lowercase letter
`isupper()`	Uppercase letter
`isalpha()`	Uppercase or lowercase letter
`isalnum()`	Uppercase or lowercase letter or a digit
`iscntrl()`	Control character
`isprint()`	Any printing character including space
`isgraph()`	Any printing character except space
`isdigit()`	Decimal digit (`'0'` to `'9'`)
`isxdigit()`	Hexadecimal digit (`'0'` to `'9'`, `'A'` to `'F'`, `'a'` to `'f'`)
`isblank()`	Standard blank characters (space, `'\t'`)
`isspace()`	Whitespace character (space, `'\n'`, `'\t'`, `'\v'`, `'\r'`, `'\f'`)
`ispunct()`	Printing character for which `isspace()` and `isalnum()` return `false`

Utilizando la Librería string

- Para utilizar la librería String hay que incluir "string.h" en el programa:

```
# include <string.h>
```

- Funciones que devuelven un puntero a "char":

  strstr (localizar un string dentro de un substring)

  strcat (encadenar)

  strncat (encadenar "n" elementos)

  strcpy (copiar)

  strncpy (copiar "n" elementos)

  strchr (localizar un elemento)

Bibliografía

- Brian Kernighan; P. J. Plauger (1976), **Software Tools**, Addison-Wesley, pp. 352, ISBN 020103669X.
- Cairó Oswaldo. **Metodología de la programación- Algoritmos, diagramas de flujo y programas-.** Ed. Alfaomega.
- Joyanes Aguilar, Luis. **Fundamentos de programación**. Ed. Mc Graw Hill.
- Joyanes Aguilar, Luis. **Programación en C.** McGraw Hill.
- Schildt, Herbert. **Lenguaje C, Programación Avanzada**. McGraw Hill.
- Bronson, Gary. **Algorithm Development and Program Design Using C**, PWS Publishing Co.; Book and Disk edition (February 15, 1996) ISBN: 0314069879.
- Standish, Thomas. **Data Structures, Algorithms, and Software Principles in C**, Addison-Wesley Pub Co; 1st edition (September 30, 1994), ISBN: 0201591189.
- Linden, Peter. **Expert C Programming**, Prentice Hall, 1994, ISBN 0131774298.
- Reek, Kenneth. **Pointers on C**. Addison Wesley, 1997. ISBN 067399866.
- Kernighan, Brian W. & Dennis, M. Richie. **El lenguaje de programación C**. 2ª Ed. Prentice Hall, 1988.
- http://zinjai.sourceforge.net/