Cadenas de Caracteres

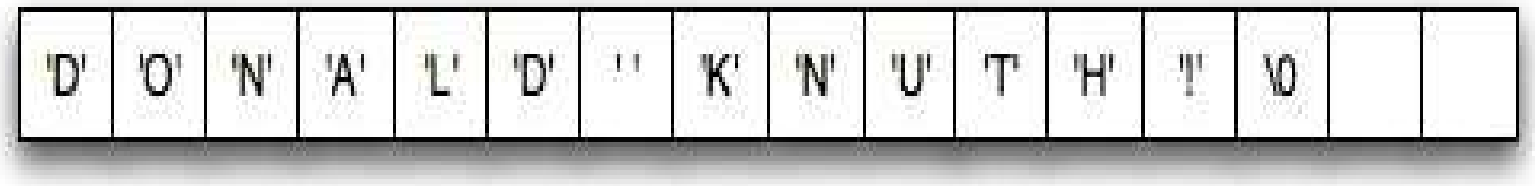
Modificaciones: Andrés Arcia
Prof. Flor Narciso
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

Cadena de Caracteres

Una cadena de caracteres (string) es un conjunto de caracteres (incluido el blanco) que se almacenan en localidades contiguas de memoria. Se representa como un vector de caracteres donde cada elemento del vector representa un carácter de la cadena.

Ejemplo

```
char nombreComputista[16];
```

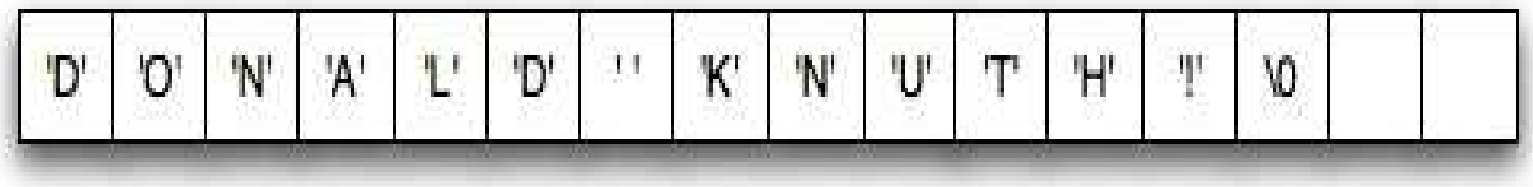


Cadena de Caracteres

Nótese que una cadena de n caracteres requerirá un vector de n+1 elementos, debido al carácter nulo ‘\0’que se añade automáticamente al final de la cadena.

Ejemplo

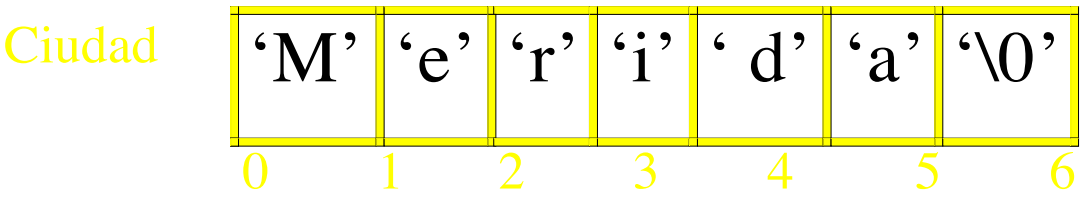
char nombreComputista[16];



Cadena de Caracteres: Ejemplo 1

Almacenar la cadena “Merida” en un vector llamado ciudad.

Nro. de Elemento	Valor del índice	Elemento del vector	Carácter de la cadena
1	0	ciudad[0]	‘M’
2	1	ciudad[1]	‘e’
3	2	ciudad[2]	‘r’
4	3	ciudad[3]	‘i’
5	4	ciudad[4]	‘d’
6	5	ciudad[5]	‘a’
7	6	ciudad[6]	‘\0’



Declaración de una Cadena de Caracteres

Notación algorítmica	*Notación en C*
cadena nombre[dim]	char nombre[dim];

donde dim = nro. de caracteres de la cadena + 1

Ejemplos:

Cadena linea[81]	char linea[81];
Cadena color[10]	char color [10];

Declaración de una Cadena de Caracteres

Ejemplo

```
#define MAXCAR 256     // Numero máximo de caracteres
                       // es 256

char  palabra[MAXCAR];
```

Se puede inicializar la cadena de caracteres al declararla:

```
char palabra[MAXCAR] = {'H', 'o', 'l', 'a', '\0'};
char palabra[MAXCAR] = "Hola";
char palabra[] = "Hola"; // Longitud maxima de la cadena
                         // es 5 (4 caracteres + '\0')
```

Acceso a una Cadena de Caracteres

Para tener acceso a los elementos de una cadena se utiliza un subíndice .

Ejemplo:

```
palabra[0] = 'H';
palabra[1] = 'o';
palabra[2] = 'l';
palabra[3] = 'a';
palabra[4] = '\0';
```

Tipo de Dato Cadena de Caracteres

Tipo de datos en C

```
#define MAXCAR 256
char  palabra[MAXCAR];
```

Tipo de datos definido por el usuario

Se utiliza la palabra *typedef*

```
#define MAXCAR 256
```

// Definicion del tipo cadena

```
typedef char cadena[MAXCAR];
```

//Declaracion

```
cadena palabra;
```

Tipos de Datos definidos por el Usuario

typedef permite a los usuarios definir nuevos tipos de datos. Una vez definido, se pueden declarar nuevas variables, arreglos, etc,, en términos de este nuevo tipo de datos.

typedef tipo_dato nombre_nuevo_tipo_dato;

Ejemplos:

```
typedef int edad;                 // Definicion del tipo de datos edad
typedef float altura;             // Definicion del tipo de datos altura

edad hembra, varon;               // Declaracion de variables
                                  // del tipo de datos edad
altura hombres, mujeres;          // Declaracion de variables
                                   // del tipo altura
```

Tipos de Datos Definidos por el Usuario

Ejemplo:

```
#define NDIAS 7
#define MAXCAR 10

typedef char cadena[MAXCAR];

cadena diasSemana[NDIAS] =
   {“lunes”,”martes”,”miercoles”,”jueves”,”viernes”,”
   sabado”,”domingo”};
```

Lectura y Escritura de Cadenas de Caracteres

Leer una cadena introducida por teclado:

```
gets(linea); // cadena completa hasta '\n'
scanf("%s", palabra); // 1 palabra
```

Escribir una cadena en la pantalla:

```
printf("%s",palabra);
puts(palabra);
```

Cadena de Caracteres: Ejemplo 2

Escribir un programa en C que lea una secuencia de caracteres ASCII y escriba una secuencia de caracteres codificados. Si el carácter es una letra o dígito, será reemplazado por el siguiente carácter en el conjunto de caracteres, excepto Z que será reemplazado por A, z por a y 9 por 0. Por tanto, 1 se transforma en 2, C en D, p en q, etc. Cualquier carácter que no sea letra o dígito será reemplazado por un punto (.).

Cadena de Caracteres: Ejemplo 3

```
#include <stdio.h>
#define MAXCAR 80



void main() {
     char linea[MAXCAR];
     int cont;

     gets(linea);
     for (cont = 0; linea[cont] != '\0'; cont++)
          if (((linea[cont] >='0') && (linea[cont] < '9')) ||
               ((linea[cont] >='A') && (linea[cont] < 'Z')) ||
               ((linea[cont] >='a') && (linea[cont] < 'z'))
                 putchar(linea[cont] + 1);
          else if (linea[cont] == '9')     putchar('0');
          else if (linea[cont] == 'Z')     putchar('A');
          else if (linea[cont] == 'z')      putchar('a');
          else    putchar('.')
}
```

Funciones para manipulación de cadenas de caracteres

Funciones de biblioteca que permiten procesar una cadena de caracteres como una entidad completa.

Para usar estas funciones es necesario incluir la biblioteca string.h

Longitud de una cadena (strlen): Devuelve la longitud de una cadena.

```
lon = strlen(palabra);
```

Asignación (strcpy): Copia la segunda cadena en la primera cadena.

```
strcpy (cadena1, cadena2); // cadena1 = cadena2
```

Funciones para manipulación de cadenas de caracteres

Comparación (strcmp): Compara dos cadenas. Si son iguales devuelve 0; si la primera es menor que la segunda devuelve un valor < 0; si la primera es mayor que la segunda devuelve un valor > 0.

```
if (strcmp(cadena1, cadena2) == 0) // Son iguales

if (strcmp(cadena1, cadena2) < 0)  // cadena1 < cadena2

if (strcmp(cadena1, cadena2) > 0)  // cadena1 > cadena2
```

Funciones para manipulación de cadenas de caracteres: Ejemplo 4.

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

#define MAXCAR 256;
typedef char Cadena[MAXCAR];

int main () {
   Cadena s1 = "hola", s2 = "adios";
   strcpy(s1, s2);
   if (strcmp(s1, s2) == 0)
       printf("Cadenas iguales\n");
   else
       printf("Cadenas diferentes\n");
return 0;
}
```

Funciones para manipulación de cadenas de caracteres: Ejemplo 6.

Defina los siguientes tipos de datos:

Un tipo Cadena capaz de almacenar cadenas de hasta 50 caracteres de longitud.

```
typedef char Cadena[51];
```

Una tipo de datos TabCad capaz de almacenar 20 cadenas de hasta 50 caracteres de longitud cada una.

```
typedef Cadena TabCad[20];
```

Arreglo de Cadenas

```
#define TAMANO_LINEA 80
#define NUMERO_ELEMS 256

int main()
{
   typedef char MiCadena[TAMANO_LINEA];
   typedef MiCadena ArrCadenas[NUMERO_ELEMS];

   ArrCadenas ac;
   int i;
   for(i=0; i<NUMERO_ELEMS; i++)
     { printf("Cadena #%i", i);
       scanf("%s",ac[i]); }
}
```

Funciones para manipulación de cadenas de caracteres: Ejemplo 7.

El siguiente procedimiento lee una cadena de caracteres, carácter a carácter, desde el teclado.

```
void leerCadena (Cadena cad) {
     int i = 0;
     char c;
     c = getchar(); // Lee un caracter del teclado
     while (c != '\n' && i < MAXCAR - 1) {
          cad[i] = c;
          i++;
          c = getchar ();
     }
     cad[i] = '\0';
}
```

Funciones de la biblioteca string.h

char *strstr(const char *, const char *)

Busca la primera ocurrencia de una cadena terminada en nulo (\0) mas pequeña en una cadena más grande.

char *strcasestr(const char *, const char *)

Similar a strstr() pero ignora mayusculas y minusculas.

size_t strlen(const char *);

Determina el tamaño de una cadena cualquiera.

char *strcpy(char *, const char *);

Copia la cadena constante del segundo parámetro en el primero.

char *strcat(char *, const char *);

Añade la cadena constante del segundo parámetro a la cadena del primer parámetro.

Funciones de la biblioteca string.h

`char * strsep(char **stringp, const char *delim);`

Separa las cadenas contenidas el stringp y cuyo delimitador es delim. Sucede que reemplaza el delimitador por el caracter de final de cadena '\0'.

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

int main()
{
char **ap, *argv[10], *inputstring, buf[80];
int i,j;

inputstring=buf;

gets(inputstring);

// Continua ...
```

Funciones de la biblioteca string.h

```
for (ap = argv, i=0; (*ap = strsep(&inputstring, "   ")) != NULL; i++)
{
    if (**ap != '\0')
        if (++ap >= &argv[10])
           break;
}

for (j=0; j<i; j++)
  printf("%s\n",argv[j]);

printf("cadena original: %s\n",inputstring);

return 0;
}
```

Ejercicios

Escribir un programa en C que lea una línea de texto, la almacene en un vector y la escriba al revés. La longitud de la línea no será especificada (terminará al pulsar la tecla *Enter*), pero se supone que no excederá de 80 caracteres.

Escribir la declaración de las siguientes variables:

- Una variable cadena *cad* con un máximo de 20 caracteres.
- La misma variable *cad* inicializada con el valor “Sevilla”.
- Dos variables *nombre* y *apellido* de tipo *Cadena*.
- Un vector *nombres* de tipo *Tabcad*.
- Un vectpr *meses* inicializado con los nombres de los doce meses del año.
- Un vector *dias_semana* inicializado con los nombres de los siete días de la semana.

Ejercicios

Escribir una función

logico busca_caracter(Cadena cad, char c)

que busque un carácter *c* en una cadena *cad*, devolviendo un valor lógico Cierto si lo encuentra y Falso en caso contrario.

Escribir una función

int busca_caracter(Cadena cad, char c)

que cuente el número de apariciones de un carácter *c* en una cadena *cad*, devolviéndolo como resultado.

Escribir una función que reciba como parámetro una cadena de caracteres y elimine los espacios en blanco del final de la cadena.

Ejercicios

Escribir un procedimiento

void dia_semana(int dia, Cadena nomdia)

que reciba como parámetro de entrada un número del 1 al 7 que representa el día de la semana (1=lunes, 2=martes, 3=miercoles, 4=jueves, 5=viernes, 6=sabado, 7=domingo) y devuelva en la cadena *nomdia* el nombre de dicho día. Utilice una tabla *dias_semana*.

Escribir un procedimiento que lea una palabra de hasta 20 caracteres y la escriba como se ve en la figura:

```
Entrada: HOLA
Salida:   HOLA
          O  L
          L  O
          ALOH
```