# **Cadenas de caracteres**

Todos los lenguajes de programación disponen de algún método para tratar las cadenas de caracteres (en inglés *string* -cadena-), ya que frecuentemente un programa tiene que trabajar con nombres, descripciones, códigos alfanuméricos, etc.

En C no hay un tipo de dato primario para las cadenas. C dispone del tipo de dato primario char, que permite almacenar 1 carácter, por lo que para almacenar un conjunto de caracteres o cadena hace falta un vector de char.

Por ejemplo, para almacenar una palabra de 10 caracteres:

**char palabra[11];**

La razón por la que hay que agregar un carácter adicional es que, para indicar el fin de una cadena, C exige un caracter “terminador” especial, el cual será luego reconocido por las funciones que tratan las cadenas. Este **caracter es el caracter nulo ‘\0’**.

Siempre hay que definir el vector de char del tamaño máximo necesario, y luego sumar 1 para el caracter terminador. Por ejemplo, si se quiere almacenar en una cadena el nombre de un mes del año, y no se sabe de antemano cuál de los meses será, habrá que analizar primero cual es el mes que más letras tiene, y luego sumarle 1. Para este caso, el vector será char mes[10]. Si se almacena un mes con menor cantidad de letras que 9, C pondrá luego del último caracter ingresado (al dar Enter) el caracter nulo, de la siguiente manera:

| ‘N’ | ‘O’ | ‘V’ | ‘I’ | ‘E’ | ‘M’ | ‘B’ | ‘R’ | ‘E’ | **‘\0’** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ‘n’ | ‘o’ | ‘v’ | ‘i’ | ‘e’ | ‘m’ | ‘b’ | ‘r’ | ‘e’ | **‘\0’** |
| ‘A’ | ‘b’ | ‘r’ | ‘i’ | ‘l’ | **‘\0’** |  |  |  |  |

Trabajar con cadenas, entonces, es trabajar con vectores de char, por lo tanto no es válida la siguiente asignación:

mes=”abril”;

Tampoco es válida la siguiente proposición lógica:

if(mes= = “Abril”)

Para este tipo de acciones, será necesario hacer funciones propias, o utilizar las funciones que C dispone para trabajar con cadenas. Todas estas funciones reciben la dirección de inicio de la cadena (el nombre del vector), y marcan su final con el carácter nulo.

Lo que si es posible es asignar cadenas literales al declarar las variables, de las siguientes maneras:

char cadena[5]=”hola”;

char cadena2[ ]=”hola”;

char\* cadena3=”hola”;

En todos los casos el compilador creará un vector de char de 5 caracteres que contendrá la cadena “hola”, y el carácter terminador ‘\0’. En el segundo caso se omitió dentro de los corchetes la dimensión, ya que cualquiera que sea el número que allí se coloque el vector se ajustará al tamaño estrictamente necesario para almacenar los caracteres que se asignen.

La última inicialización, en la que se usa un puntero, puede ser tratada por algunos compiladores como constante, de manera tal que su modificación no será permitida. No debe utilizarse entonces si en alguna parte del código es necesario modificar el contenido de la cadena.

# 

# 

# **Funciones de cadenas**

# **Funciones de entrada y salida**

**Entrada en C**

**scanf(“%s”, mes);//**mes es un vector de char previamente declarado

capturará los caracteres que se ingresen por teclado hasta que se presione la tecla Enter. Al encontrar el valor correspondiente a la tecla Enter, se dará por concluida la entrada y se colocará automáticamente el caracter nulo al final de la cadena.

**scanf()** no acepta espacios en blanco. Si se ingresa un espacio en blanco, por ejemplo para separar un nombre de un apellido, pondrá el terminador en el lugar del espacio en blanco. Está incluida en stdio.h (o cstdio)

**gets(mes);**

gets() requiere como parámetro la dirección de inicio de la cadena (esto es el nombre del vector de char declarado). Acepta espacios en blanco como parte de la cadena. Está incluida en stdio.h (o cstdio). No controla la cantidad de caracteres ingresados.

**Salida**

**printf(“%s”,mes)**

printf() al recibir el especificador “%s” interpreta que el paramétro que le sigue a la coma es una cadena, y muestra todos los caracteres que hay desde la dirección mes, hasta que encuentra el caracter nulo.

**puts(mes);**

puts() requiere como parámetro la dirección de inicio de la cadena. También puede utilizarse puts para imprimir una cadena definida directamente dentro de los paréntesis. Por ejemplo:

**puts(“Este es un cartel que se imprimirá en la pantalla”);**

Para definir una cadena, el texto debe ponerse entre comillas dobles.

puts() está incluído en stdio.h (o cstdio).

## **Funciones de la librería string.h (cstring si compilamos en C++)**

Algunas de las funciones de uso más frecuente son:

**strcpy(cadena1, cadena2);**

strcpy() copia el contenido de la cadena2 en la cadena1, y es la función que debe utilizarse para asignar una cadena a otra cadena. El parámetro cadena2 puede ser un vector de char, o una cadena literal.

**int strcmp(cadena1, cadena2)**

strcmp() se utiliza para comparar si una cadena es “mayor”, “menor’’ o “igual” que otra. Que una cadena sea mayor o menor que otra, se evalúa de acuerdo con el orden lexicográfico de las cadenas, es decir, su posición relativa en el diccionario.

La función devuelve los siguientes valores:

- 0, si ambas cadenas son iguales.

- <0, si cadena1 es menor que cadena2.

- >0, si cadena1 es mayor que cadena2.

**int strcmpi(cadena1, cadena2)**

Tiene la misma funcionalidad que strcmp, con la diferencia que una palabra en mayúsculas será considerada igual que una en minúsculas.

**Otras funciones:**

**int strlen(char \*)**: devuelve la cantidad de caracteres de la cadena, sin contar el caracter nulo.

**strcat(cadenafinal, cadena):** agrega a cadenafinal el contenido de cadena.

**strlwr(cadena):** convierte todos los caracteres de cadena a minúsculas.

**strupr(cadena):** convierte todos los caracteres de cadena a mayúsculas.

## **Operadores de C++ para entrada/salida de cadenas**

Además de las funciones, se puede utilizar para el ingreso y la salida por pantalla

cin>> y cout<<.

Debe tenerse en cuenta que cin>> no admite espacios. Al encontrar un espacio cin colocará el carácter nulo.

Para almacenar cadenas con espacios puede usarse cin.getline(char \*cad, int longitud), o cin.get(char \*cad, int longitud).

Estas funciones tienen como parámetros la cadena sobre la que se quiere escribir, más la longitud de la cadena. Si se ingresan más caracteres que los definidos en el parámetro longitud serán ignorados. Por ejemplo:

int main()

{

char cadena1[5];

cin.getline(cadena1,4); //sólo se aceptarán 4 caracteres

cout<<cadena1;

return 0;

}

Nota: cin.getline() o cin.get() pueden no funcionar si antes de su ejecución hubo otra instrucción que requirió que se presione la tecla Enter. Para evitar este problema se puede anteponer a cin.getline() la instrucción cin.ignore(), cin.get() (sin parámetros) o fflush(stdin).