Elementos de Programación

UNIDAD 8. CADENAS DE CARACTERES (STRINGS)

INDICE

UNIDAD 8 - Cadenas de caracteres (strings)

OBJETIVOS: *Comprender la forma del manejo de texto en el lenguaje C. Utilizar funciones para trabajar con texto. Incorporar el manejo de texto a los programas existentes.*

1. Introducción

El lenguaje C, a diferencia de otros lenguajes, hace un manejo muy particular de las variables para almacenar texto. Un texto no es más que una sucesión de letras, por lo tanto, se necesita almacenar varios caracteres uno a continuación del otro. Una variable del tipo char permite almacenar un único carácter, por lo tanto, para almacenar un texto se debe definir un vector de caracteres con tantas posiciones como letras pueda tener el texto que se quiere almacenar. Entonces, en un principio se podría decir que una variable del tipo string o cadena de caracteres, no es más que un vector de char. Sin embargo, esta afirmación no es del todo cierta ya que hay una característica que las diferencia. Toda cadena de caracteres define su terminación con un carácter de control '\0'. Este carácter especial indica el fin de la cadena.

Por ejemplo, si se quiere ingresar un nombre, no se sabe exactamente el largo del mismo por lo tanto se define un vector con una cantidad suficiente de acuerdo al nombre más largo que se quiera guardar, por ejemplo:

```
char nombre [20];
```

Declara un vector de char de 20 posiciones. Si en dicho vector se almacena un nombre corto, por ejemplo "JUAN", dicho nombre solo ocupará 4 de los 20 caracteres, y por lo tanto, luego de la letra N se debe poner el carácter de fin de cadena indicando hasta donde llega dicho nombre.

J	U	A	N	\0															

Entonces la diferencia principal entre un vector de caracteres y un string es la presencia de ese carácter especial de fin de cadena.

2. Lectura por teclado de strings

Ahora bien, como se indicó, un string es un vector de caracteres, pero al momento de solicitar el ingreso por teclado del dato no se le va a pedir al usuario que ingrese una a una las letras en cada posición del vector. Para la lectura de string el lenguaje dispone de distintas alternativas:

2.1 scanf

Tradicionalmente todos los ingresos desde teclado hasta ahora se realizaron utilizando esta función. Donde se pone como primer parámetro el formato de los datos a ingresar y como segundo parámetro la dirección de memoria donde almacenar el dato. El formato para los strings es %s. Veamos el siguiente ejemplo:

```
#include <stdio.h>
int main()
{
      char nombre[20];
      printf ("Ingrese un nombre: ");
      scanf("%s", nombre);
      printf ("El nombre ingresado es: %s", nombre);
      return 0;
}
```

Note que en el scanf no fue necesario poner el & para obtener la dirección de memoria ya que al ser nombre un vector contiene la dirección de inicio del mismo. Por lo tanto, el & NO debe ponerse ya contiene la dirección de memoria que se necesita para indicarle donde guardar el dato.

Si se ejecuta y se prueba este programa e ingresa por teclado JUAN y luego presiona enter, automáticamente se completa el vector guardando la J en la posición 0, la U en la posición 1, y así sucesivamente, hasta la última letra y automáticamente en la posición siguiente a la letra N el scanf agrega el carácter de fin de cadena \0 ya que se le indica que está leyendo un string con el formato %s.

Hasta aquí no hay problemas y todo funciona normalmente. Pero el scanf con formato de string tiene un problema. Si prueba nuevamente el ejemplo y en el nombre ingresa JUAN CARLOS, en pantalla mostrará:

```
El nombre ingresado es: JUAN
```

Internamente en el vector solo guardará JUAN y a continuación el carácter de fin de cadena. Esto se debe a que el espacio corta el ingreso del scanf almacenando solo JUAN y haciendo que se pierdan el resto de los datos. Recordemos que cuando con scanf se leía más de una variable una forma de separarlas era con espacio y por ello no funciona correctamente en la lectura de string.

2.2 gets

En la biblioteca stdio.h hay una función llamada gets que permite solucionar el problema del scanf al leer un string. Reemplazando dicha función en el programa anterior quedaría:

```
#include <stdio.h>
int main()
{
      char nombre[20];
      printf ("Ingrese un nombre: ");
      gets(nombre);
      printf ("El nombre ingresado es: %s", nombre);
      return 0;
}
```

De esta forma se elimina el problema del scanf y si se ingresa por teclado JUAN CARLOS en el vector los datos quedarán almacenados de la siguiente forma:

J	U	A	N		C	A	R	L	O	S	\0								

Y en pantalla se mostrará:

```
El nombre ingresado es: JUAN CARLOS
```

2.3 fgets

Tanto el scanf como gets NO CHEQUEAN LIMITES, por lo tanto, si se define el vector para guardar 20 caracteres y se ingresa un nombre más largo la variable se guarda en las posiciones siguientes que NO tiene reservadas haciendo que probablemente se pierdan datos de otras variables, y por lo tanto, haciendo que el programa funcione mal. Algunos compiladores darán una advertencia diciendo que es peligroso utilizar el gets debido a que se puede escribir memoria no reservada.

Para solucionar esto se puede acudir a otra función que se encuentra también en la biblioteca stdio.h. Esta función es fgets y tiene 3 parámetros, el primero es el vector donde se va a guardar el dato, el segundo la cantidad de caracteres que se pueden almacenar en dicho vector y el tercero es de donde se leen los datos que en este caso siempre será desde teclado. La siguiente línea:

```
fgets(nombre,20, stdin);
```

Indica que se va a leer desde teclado (indicado por stdin), como máximo 20 caracteres y se va a guardar a partir de la dirección de memoria indicada por nombre.

Reemplazando en el programa anterior:

```
#include <stdio.h>
int main()
{
	char nombre[20];
	printf ("Ingrese un nombre: ");
	fgets(nombre,20,stdin);
	printf ("El nombre ingresado es: %s", nombre);
	return 0;
}
```

Al probar este programa si se ingresa un texto de más de 20 caracteres, guardará en el vector los 19 primeros y en la última posición pondrá el carácter de fin de cadena \0, el resto de caracteres no los guarda, y por lo tanto, no sobrescribe memoria no reservada.

Al parecer esta función soluciona todo los problemas pero si se prueba nuevamente ingresando por teclado JUAN CARLOS y luego la tecla enter, se verá que el vector almacena lo siguiente:

J	U	A	N		C	A	R	L	O	S	\n	\0							

Es decir, que la función fgets almacena también el enter ingresado. Por lo tanto, cuando se muestra la cadena luego de JUAN CARLOS dejará un salto de línea. Dependerá del uso que quiera darle al string si ese salto de línea molesta o no. Una posible solución luego de leer el string con fgets, sería buscar si quedó almacenado el \n y eliminarlo poniendo en su lugar el \0. En el siguiente ejemplo ingreso del string se realiza mediante una función LeerTexto que lee mediante fgets y luego busca y elimina el salto de línea almacenado:

```
#include <stdio.h>
 void LeerTexto (char[], int);
 int main()
 {
	char nombre[20];
	printf ("Ingrese un nombre: ");
	LeerTexto(nombre,20);
	printf ("El nombre ingresado es: %s", nombre);
	return 0;
 }

 void LeerTexto (char texto[], int largo)
 {
	int i;
	fgets(texto, largo, stdin);
	i=0;
	while (texto[i]!='\0')
	{
		if (texto[i]=='\n')
			texto[i]='\0';
		else
			i++;
	}
 }
```

3. Mostrar strings por pantalla

En los ejemplos anteriores se vio que es posible mostrar los string mediante un printf con formato %s, esta es la forma más difundida y es la que más utilizará ya que permite darle formato, mostrar otras variables, etc.

Otra forma para mostrar un string es mediante la función puts disponible en la biblioteca stdio.h. Esta función recibe como parámetro el string y lo muestra por pantalla y además automáticamente luego de mostrar el texto hace un salto de línea.

Es decir, que la siguiente línea de código (definiendo nombre como un string):

```
puts(nombre);
```

es equivalente a:

```
printf ("%s\n", nombre);
```

4. Inicialización de Strings

Al declarar la memoria del string es posible asignar un valor, para ello se le asigna una constante del tipo string que se escribe entre comillas. El carácter de fin de cadena se agrega automáticamente luego de la última letra. A continuación, observe algunos ejemplos de inicialización y como queda el vector resultante:

```
char nombre[20] = "ANA MARIA";
```

A	N	A		M	A	R	I	A	\0										

```
char nombre[] = "ANA MARIA";
```

A	N	A		M	A	R	I	A	\0

Si no se especifica el tamaño automáticamente reserva la memoria mínima suficiente para almacenar el texto y el carácter de fin de cadena.

¡Cuidado!! Si se le pone un tamaño al vector y se le asignan más caracteres de los definidos NO guardará el \0, y por lo tanto, ya no puede tratarse como un string ya que la función para mostrar no encontrará el fin de cadena.

```
char nombre[3] = "MARCELO";
```

M	A	R

5. Biblioteca para el manejo de texto (string.h)

Debido a que los string son vectores con un carácter especial que indica el final de la cadena hay ciertas cosas como copiar uno a otro, comparar, etc que NO se pueden hacer directamente ya que no son un tipo de dato en sí mismos. Identificando el fin de cadena es posible armar distintas funciones para hacer esas tareas. Por ejemplo, para saber la cantidad de caracteres de un string se puede recorrer posición a posición el vector hasta encontrar el fin de cadena contando cuantas posiciones nos desplazamos. Para copiar un string en otro es similar a la copia de vectores donde

se copia posición a posición en este caso hasta encontrar el fin de cadena. Todas estas funciones las podemos desarrollar, pero dentro del lenguaje C ya existe una biblioteca que maneja distintas funciones relacionadas con cadenas de caracteres. La biblioteca string.h

Dentro de string.h hay muchas funciones, veremos algunas de ellas:

strlen	Determina la longitud de una cadena
strcpy	Copia una cadena a otra
strcat	Concatena dos cadenas dejando el resultado en la primera
strcmp	Compara dos cadenas
strcmpi	Compara dos cadenas ignorando si son mayúsculas o minúsculas

Función strlen:

Determina la longitud de una cadena, sin contabilizar el carácter nulo de terminación de la misma.

Sintaxis: strlen (cadena)

La función retorna un entero indicando la cantidad de caracteres.

Ejemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
	char cadena [21];
	printf ("Ingrese una cadena de no más de 20 caracteres \n");
	gets (cadena);
	printf ("La cadena ingresada contiene: %d caracteres", strlen(cadena) );
	return 0;
}
```

Función strcpy:

Copia desde una cadena de origen hacia una cadena de destino.

Sintaxis: strcpy (cadena destino, cadena origen)

Ejemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
	char original [15], copia [15];
	printf ("Ingrese una cadena que sera luego copiada \n");
	gets (original);
	strcpy (copia, original);
	printf ("La cadena original es: %s y la copia es: %s", original, copia );
	return 0;
}
```

También strcpy se puede usar para asignar una constante a un string cuando no se hace al momento de inicializar la variable.

```
#include <stdio.h>
#include <string.h>
int main ()
{
	char texto [25];
	strcpy(texto, "MENSAJE");
```

```
            puts(texto);
            return 0;
      }
```

Función strcat:

Concatena (añade) una cadena detrás de otras quedando el resultado en la cadena que se encuentra en primer orden.

Sintaxis: strcat (cadena receptora, cadena a añadir)

Ejemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
      char receptor [40] = "Se agrego lo siguiente", dador [] =", me agregue";
      printf ("Las cadenas por separado son: \n\t %s\n\t %s", receptor, dador);
      strcat (receptor, dador);
      printf ("\nLas cadenas unificadas son: \n\t %s ", receptor );
      return 0;
}
```

Es importante asegurarse de que la cadena receptora tenga el espacio suficiente para guardar la cadena a añadir caso contrario sobrescribe memoria no asignada

Función strcmp:

Esta función compara dos cadenas y devuelve el resultado de la comparación.

Sintaxis: strcmp (cadena 1, cadena 2)

El valor que devuelve que será el resultado de la comparación es el siguiente:

Si las cadenas son iguales	devolverá un cero (0)
Si la cadena 1 es mayor que la cadena 2	devolverá un valor positivo
Si la cadena 1 es menor que la cadena 2	devolverá un valor negativo

Ejemplo:

```
#include <stdio.h>
#include <string.h>
int main ()
{
      char cadena1[30], cadena2[30];
      printf ("Ingrese la primer cadena:");
      gets(cadena1);
      printf ("Ingrese la segunda cadena:");
      gets(cadena2);
      if (strcmp (cadena1, cadena2) == 0)
            printf ("\nAmbas cadenas son iguales " );
      else
          if (strcmp (cadena1, cadena2) > 0 )
              printf ("\nLa cadena1 es mayor que la cadena2");
          else
              printf ("\nLa cadena2 es mayor que la cadena1");
      return 0;
}
```

La comparación NO es por largo sino alfabética, comparando el peso en valor ASCII de cada una de las letras, posición a posición, hasta encontrar alguna diferencia. Hay que recordar que el ASCII de

las minúsculas es mayor que el de las mayúsculas, por lo tanto, una palabra en minúsculas será mayor que una en mayúscula. A continuación, se muestran algunos ejemplos:

Cadena1	Cadena2	Mayor / iguales
HOLA	HOLA	IGUALES
Hola	hola	Cadena2
hola mundo	hola	Cadena1
ana	anana	Cadena2
teXto	texto	Cadena2

Función strcmpi:

Es exactamente igual a la función strcmp pero ignora si son mayúsculas o minúsculas es decir que el string "ANA" es igual al string "ana" y también es igual al string "anA" o cualquiera de sus variantes.

6. Vector de strings

Un string es en realidad un vector de caracteres, ahora si se quiere guardar varios string en un vector entonces tiene que definir varios vectores de caracteres y para ello se utiliza una matriz. Por lo tanto, un vector de strings es una matriz de caracteres donde en cada fila se guardará un string finalizado con \0. Si bien se define como una matriz se manejará tanto para la lectura como para mostrar como un vector ya que no se va a recorrer letra a letra, sino que se va a leer y mostrar las palabras completas.

Para definir un vector de string entonces se define una matriz donde:

La cantidad de filas es la cantidad de string que se quiere guardar y la cantidad de columnas indica la cantidad máxima de caracteres de cada uno de los strings (contando el \0).

El siguiente esquema representa un vector de string, donde se pueden almacenar 5 nombres de hasta 10 caracteres cada uno (se define de 11 para el \0).

char nombres [5][11];

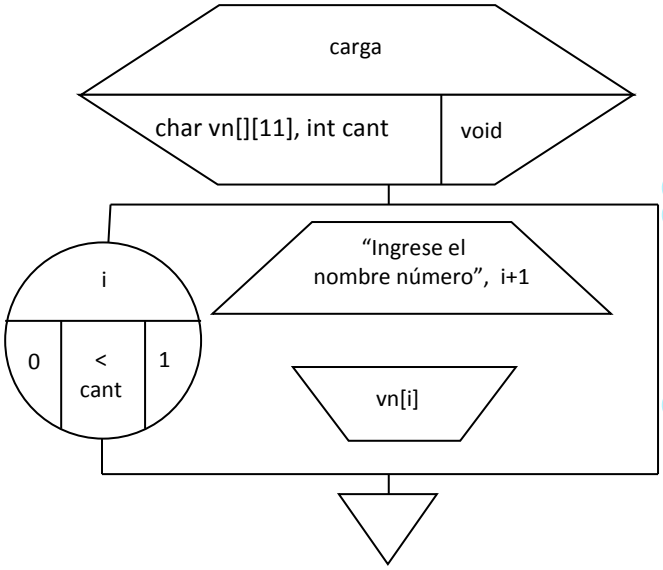
A	N	A	\0							
L	U	I	S	\0						
J	U	A	N		P	A	B	L	O	\0
L	U	C	I	A	N	O	\0			
M	A	R	I	A	\0					

Los nombres ingresados en cada fila pueden ser de cualquier largo siempre y cuando no pasen la cantidad máxima de caracteres reservada.

7. Funciones sobre vectores de string

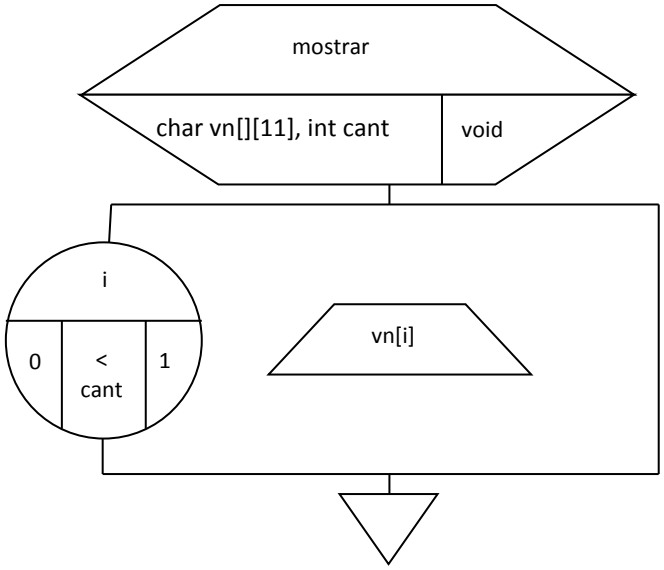
Como se indicó anteriormente si bien un vector de string es una matriz se maneja como un vector. A continuación, se desarrollan algunas de las funciones más usadas. Tomando cadenas de 10 caracteres máximo.

7.1 Carga



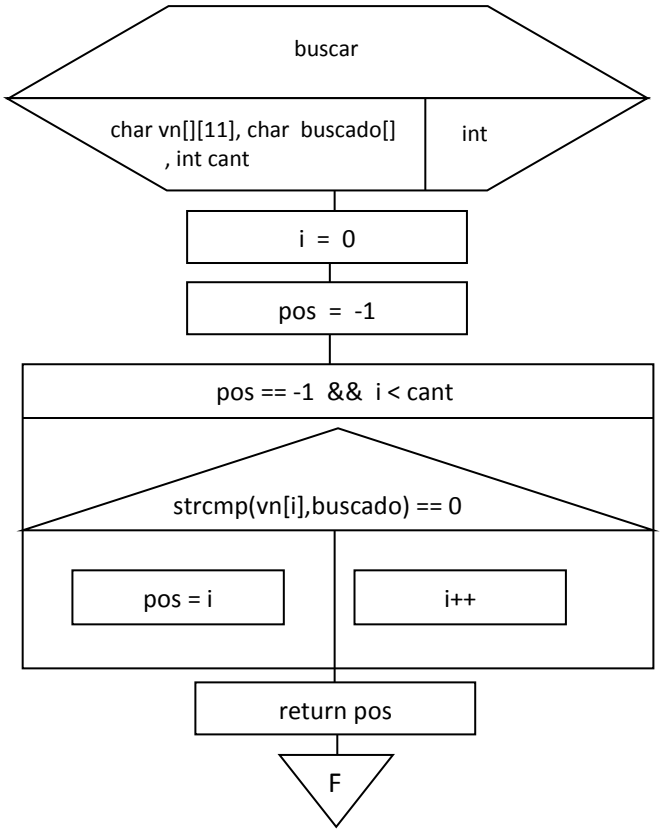

```
void carga(char vn[][11],int cant)
{
   int i;
   for (i=0;i<cant;i++)
   {
      printf ("Ingrese el nombre numero %d: ", i+1);
      gets(vn[i]);
   }
}
```

7.2 Mostrar




```
void mostrar(char vn[][11],int cant)
{
      int i;
      for (i=0;i<cant;i++)
            puts(vn[i]);
}
```

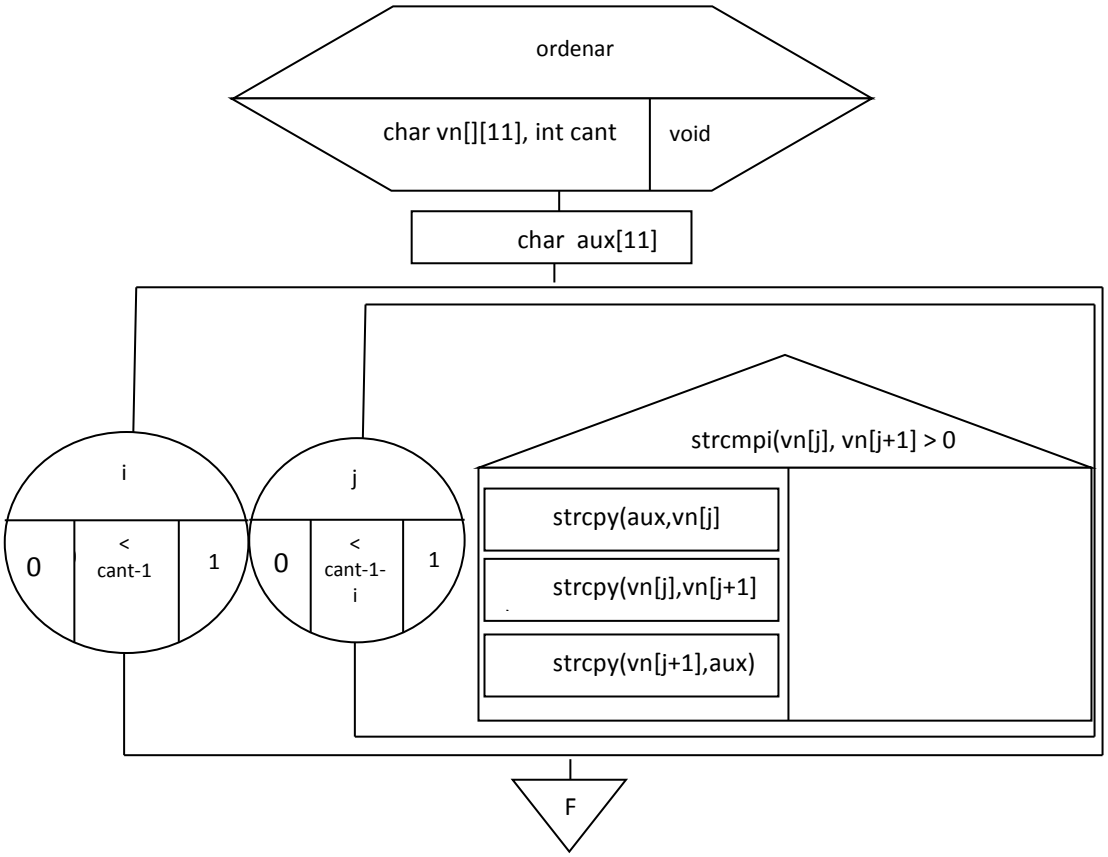
7.3 Búsqueda Secuencial




```
int buscar(char vn[][11],char
buscado[], int cant)
{
   int i=0, pos=-1;
   while (pos==-1 && i<cant)
   {
      if (strcmpi(vn[i], buscado)==0)
            pos =i;
      else
            i++;
   }
   return pos;
}
```

7.4 Orden

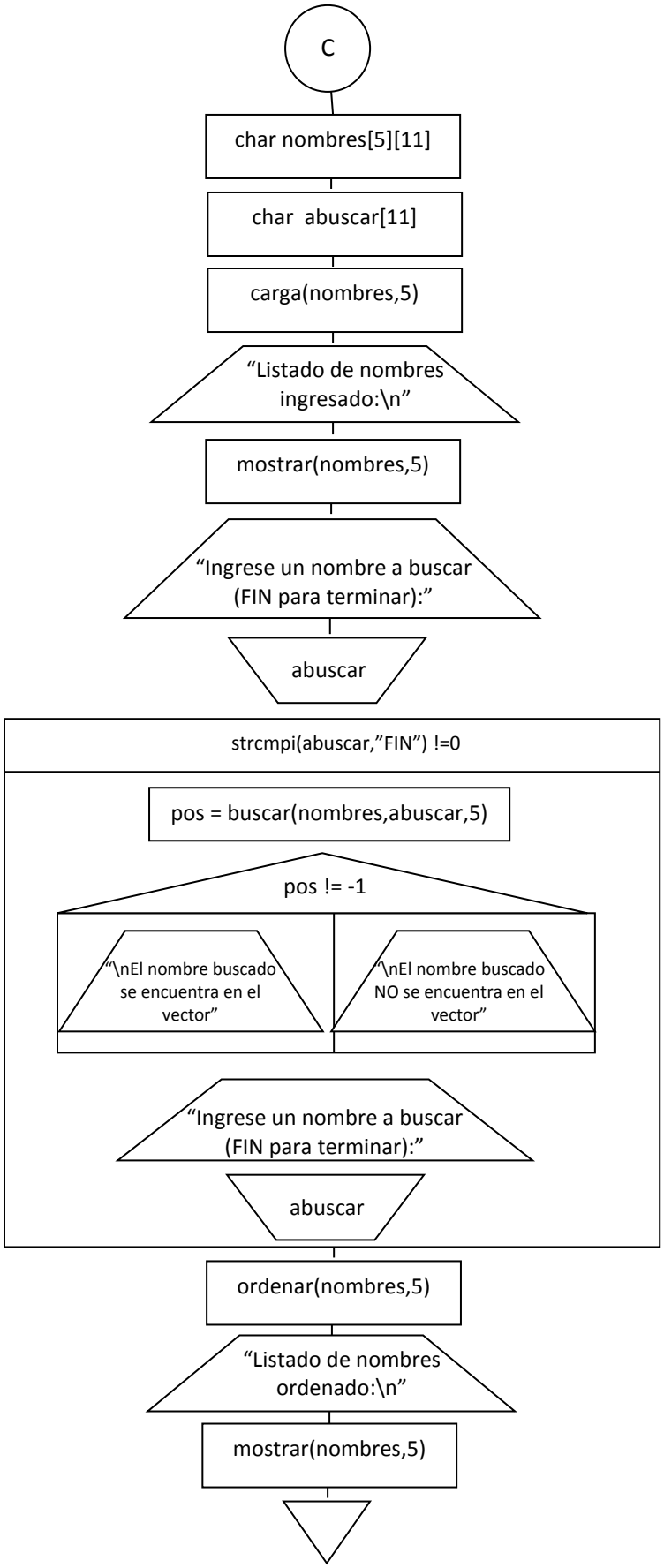
```
void ordenar (char vn[][11],int cant)
{
      int i,j;
      char aux[11];
      for (i=0;i<cant-1;i++)
      {
            for (j=0;j<cant-1-i;j++)
            {
                  if (strcmpi(vn[j],vn[j+1]) > 0)
                  {
                        strcpy(aux, vn[j]);
                        strcpy (vn[j],vn[j+1]);
                        strcpy (vn[j+1], aux);
                  }
            }

      }
}
```



8. Ejemplo de aplicación

Utilizando las funciones definidas anteriormente se desea realizar el siguiente programa, cargar 5 nombres de hasta 10 caracteres cada uno en un vector de string. Luego ingresar nombres por teclado e indicar si se encuentran en el vector, la búsqueda finalizará con un nombre igual a “FIN”. Luego mostrar el listado de nombres ordenado alfabéticamente de menor a mayor.

No se desarrollan las funciones ya que son las mismas del punto 7.


C
char nombres[5][11]
char abuscar[11]
carga(nombres,5)
“Listado de nombres ingresado:\n”
mostrar(nombres,5)
“Ingrese un nombre a buscar (FIN para terminar):”
abuscar
strcmpi(abuscar,”FIN”) !=0
pos = buscar(nombres,abuscar,5)
pos != -1
“\nEl nombre buscado se encuentra en el vector”
“\nEl nombre buscado NO se encuentra en el vector”
“Ingrese un nombre a buscar (FIN para terminar):”
abuscar
ordenar(nombres,5)
“Listado de nombres ordenado:\n”
mostrar(nombres,5)

```
#include <stdio.h>
#include <string.h>
void carga(char[][11],int);
void mostrar(char[][11],int);
int buscar(char[][11],char [], int);
void ordenar (char[][11],int);
int main ()
{
    int pos;
    char nombres[5][11], abuscar[11];
    carga(nombres,5);
    printf("\nListado de nombres ingresado:\n");
    mostrar(nombres,5);
    printf ("\nIngrese un nombre a buscar (FIN para terminar): ");
    gets(abuscar);
    while (strcmpi(abuscar, "FIN")!=0)
    {
        pos = buscar(nombres,abuscar,5);
        if (pos!=-1)
            printf ("\nEl nombre buscado se encuentra en el vector");
        else
            printf ("\nEl nombre buscado NO se encuentra en el vector");

        printf ("\nIngrese un nombre a buscar (FIN para terminar): ");
        gets(abuscar);
    }
    ordenar(nombres,5);
    printf("\n\nListado de nombres ordenado:\n");
    mostrar(nombres,5);
    return 0;
}
```