UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

CADENA DE CARACTERES

Cadenas de Caracteres

2013
Transversal de Programación Básica
Proyecto Curricular de Ingeniería de Sistemas

Objetivos

- Conocer las propiedades de los arreglos de caracteres así como las diferentes operaciones que pueden ser realizadas en ellos.
- Solucionar problemas utilizando arreglos de caracteres

Introducción

Los elementos del tipo carácter (tipo **char** en lenguaje C) se pueden agrupar para formar secuencias que se denominan cadenas de caracteres, o simplemente cadenas. En la memoria del computador una cadena se guarda en un arreglo de tipo carácter.
Un arreglo es una estructura de datos, o más técnicamente, un espacio de memoria que permite almacenar una colección de elementos, todos del mismo tipo. Conviene imaginar un arreglo como una secuencia contigua de celdas (espacios de memoria), o ***casillas***, en cada una de las cuales se puede guardar un elemento de la colección.

1. Definición

Los elementos del tipo **caracter** (tipo **char** en lenguaje **C**) se pueden agrupar para formar secuencias que se denominan **cadenas de caracteres**, o simplemente **cadenas**. En este texto, y también en el texto de un programa en **C**, las cadenas se delimitan por dobles comillas.

Por ejemplo, "**CH?*$A7!**" y "**soy cadena**" son dos cadenas, la primera formada por 8 caracteres y la segunda por 10. En la memoria del computador una cadena se guarda en un arreglo de tipo **caracter**, de tal manera que cada símbolo de la cadena ocupa una casilla del arreglo. Sin embargo, se utiliza una casilla adicional del arreglo para guardar un carácter especial que se llama **terminador de cadena**. En **C** y en el seudo-lenguaje este carácter especial es ´**\0**´.

Como lo indica su nombre, la función de este carácter especial es indicar que la cadena termina. Las dos cadenas mencionadas arriba, se representan en la memoria del computador como lo indica la siguiente figura:

´C´	´H´	´?´	´*´	´$´	´A´	´7´	´!´	´\0´
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

´s´	´o´	´y´	´ ´	´c´	´a´	´d´	´e´	´n´	´a´	´\0´
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

La ***longitud*** de una cadena se define como el número de símbolos que la componen, sin contar el terminador de cadena. Es muy importante tener en cuenta que aunque el terminador de cadena no hace parte de la cadena, sí ocupa una casilla de memoria en el arreglo.

Dado que la representación de cadenas es mediante arreglos de tipo **carácter**, aplican todos los conceptos que ya explicaron para esta estructura de datos. Por otra parte, por el hecho de que las cadenas son de uso muy frecuente y generalizado en programación, en muchos lenguajes se dispone de muchas operaciones (funciones) sobre estas. La siguiente es una lista corta de estas operaciones para el seudolenguaje, en la cual debe suponerse que **cad**, **cad1** y **cad2** son nombres de arreglos de tipo **carácter**.

Operación	Descripción
leerCadena(cad)	Guarda la cadena digitada por el usuario en el arreglo **cad**
escribirCadena(cad)	Escribe en la pantalla la cadena **cad**
longitudCadena(cad)	Retorna la longitud de la cadena **cad**
copiarCadena(cad1, cad2)	Copia la cadena **cad2** en la cadena **cad1**
concatenarCadena(cad1, cad2)	Retorna la concatenación de **cad1**con **cad2**, en la cadena **cad1**
compararCadena(cad1, cad2)	Retorna menos uno (-1) si **cad1** es menor que **cad2**, cero (0) si son iguales y uno (1) si **cad2** es menor que **cad1**

Para determinar si una cadena es menor que otra se usa el orden lexicográfico, es decir, el mismo que usan los diccionarios. Así por ejemplo, “casa” es menor “casita”, y esta a su vez es menor que “caza”. De otra parte, la operación ***leerCadena*** pone automáticamente el terminador de cadena, lo mismo que las operaciones ***copiarCadena*** y ***concatenarCadena***.

Ejemplos

A continuación se muestran algunos ejemplos con el objeto de esclarecer la teoría antes presentada

- **Ejemplo Uno**

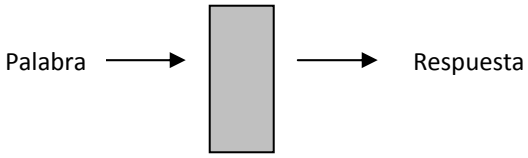
Un ***palíndromo*** es una palabra o frase, si se quiere del idioma español, que se puede leer igual de izquierda a derecha y de derecha a izquierda, obviando signos de puntuación y espacios. Para aclarar, son palíndromas las siguientes frases y palabras:

- Anilina
- Amor a Roma
- Dábale arroz a la zorra el abad
- Reconocer
- Anita lava la tina
- Ala

Considere el problema de construir un algoritmo que lea una PALABRA, de longitud máxima 30, y determine si es palíndromo o no.

- Las entradas (datos conocidos) para el algoritmo son: La palabra a considerar
- La salida esperada (dato desconocido) es: Un mensaje que indica si es palíndromo o no

La siguiente gráfica resume las entradas y las salidas del algoritmo que se pretende diseñar. Además bautiza las variables principales:



Las condiciones iniciales y finales se pueden expresar mediante dos cláusulas: REQUIERE y GARANTIZA, de la siguiente manera:

> REQUIERE: Una palabra, llamada **pal**, que puede tener hasta 30 caracteres
> GARANTIZA: Escribe en pantalla un mensaje, **respuesta**, que indica si **pal** es palíndromo o no

Una primera versión del algoritmo solución puede ser simplemente la siguiente:

> **Inicio**
> **Paso 1. Leer la palabra**
> **Paso 2. Determinar si es palíndromo**
> **Paso 3. Escribir en pantalla el mensaje apropiado**
> **Fin**

Los pasos 1 y 3 son interacciones con el usuario que se implementan usando las operaciones sobre cadenas disponibles. La versión inicial se puede refinar detallando estos pasos y además, definiendo las variables necesarias para hacerlo:

> **Procedimiento principal**
> **Variables**
> pal: arreglo [31] de caracter
> respuesta: arreglo [20] de caracter
> **Inicio**
> escribir(“Por favor digite una frase de máximo 30 letras”)
> leerCadena(pal)
> **Paso 2. Determinar si *pal* es palíndromo**
> **Paso 3. Escribir en pantalla el mensaje apropiado**
> **Fin**

Note que el arreglo **pal** se define de dimensión 31. Puede explicar cuál es el propósito de esto?.

La parte nuclear de la solución es el Paso 2 (determinar si el arreglo **pal** contiene un palíndromo). Para esto se deben comparar el primer carácter con el último, el segundo con el penúltimo, y así sucesivamente.

En seudo-lenguaje, este proceso puede ser:

```
i:= 0                                   // i señala el primer caracter de la cadena
j:= longitudCadena(pal)-1               // j señala el último caracter de la
cadena
siga:= verdadero                        //variable que indica cuándo parar el proceso
mientras (i<j y siga) hacer
        si (pal[i]=pal[j]) entonces
              i:= i+1
              j:= j-1
        sino
              siga:= falso
        fin-si
fin-mientras
si i<j entonces
        copiarCadena(respuesta,"no es palindromo")
sino
       copiarCadena(respuesta,"es palindromo")
fin-si
```

El arreglo **respuesta** queda con la respuesta apropiada para el usuario.

El algoritmo completo se presenta enseguida. Se han definido algunas constantes para permitir que el programa sea más fácilmente modificable.

Procedimiento principal

```
Constantes
N 30                    // N es la máxima longitud de la palabra dada por el usuario
Variables
pal: arreglo [31] de caracter
respuesta: arreglo [20] de caracter
i,j: entero
siga: booleano
Inicio
escribir("Por favor digite una palabra de máximo 30 letras")
leerCadena(pal)
i:= 0                                   // i señala el primer carácter de la cadena
j:= longitudCadena(pal)-1               // j señala el último carácter de la cadena
siga:= verdadero            //variable "bandera" que indica cuándo parar el proceso
mientras (i<j y siga) hacer
      si (pal[i]=pal[j]) entonces
            i:= i+1
            j:= j-1
      sino
            siga:= falso
      fin-si
fin-mientras
si i<j entonces
```

```
        copiarCadena(respuesta,”no es palindromo”)
sino
        copiarCadena(respuesta,”es palindromo”)
fin-si
escribirCadena(respuesta)
Fin
```

Problema complementario para clase

Transcriba el código del ejemplo anterior a lenguaje C++.

Ejercicios para desarrollar en clase

1. Construya un algoritmo que lea una FRASE, de longitud máxima 30, y determine si es palíndromo o no. El algoritmo debe procesar correctamente frases de más de una palabra.

2. Construya un algoritmo que lea una frase del español de máximo 100 caracteres y determine cuántas palabras, vocales y consonantes tiene.

3. Construya un algoritmo que lea dos palabras del español y determine si la primera es sufijo de la segunda. Por ejemplo, lote es prefijo de casalote.

4. Construya un algoritmo que lea dos palabras del español y determine si la primera es parte de la segunda. Por ejemplo, alo es parte de casalote.

Ejercicios para desarrollar en casa

1. Suponga que la operación longitudCadena no está disponible. Escriba un algoritmo que calcule la longitud de una cadena, bajo el supuesto de que el terminador de cadena aparece en alguna casilla del arreglo.

2. Suponga que la operación compararCadena no está disponible. Escriba un algoritmo que compare dos cadenas, bajo el supuesto de que ambas tienen el terminador de cadena.

3. Escriba un algoritmo que invierta una cadena. Por ejemplo, si la cadena es épica, su inversa es acipé.

Anexos

En esta sección se mostrara como es la codificación de la teoría vista en clase, para ello se utilizara como herramienta de codificación: C++.

Codificación en C++ de arreglos y matrices

	Seudocódigo	C++
Cadena	<NOMBRE> : arreglo [<N>] de caracter	char <NOMBRE>[<N>];

Referencias

Lectura de Profundización

- http://ocw.unican.es/ensenanzas-tecnicas/fundamentos-de-informatica/Curso-Fortran-6.pdf
- http://www.slideshare.net/wladimirclipper/cadena-caracteres

Imágenes:

Las imágenes utilizadas en este documento fueron tomadas de www.google.com

Fuentes:

- http://www.diatel.upm.es/cgonzalez/presentaciones/cadenas_de_caracteres.pdf
- http://msanchez.usach.cl/lcc/computacion/cadena-caracteres.pdf
- http://www.tonahtiu.com/notas/estructuras/alto_constantes_caracter.htm
- https://info1utnfrc.wikispaces.com/file/view/Apunte+Cadenas+de+Caracteres.pdf