Taller Preparatorio Primer Parcial

Luis Garreta

Laboratorio de Programación
Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

1. Funciones de manejo de cadenas

Implemente las siguientes funciones de manejo de cadenas

Función	Definición	Ejemplo
int longCadena (char *cad)	Retorna la longitud de la cadena de entrada "cad"	char s="Cadena demo"; n = longCadena (s) printf("La longitud de s es: %d\n", n);
void copiarCadenas (char *cadOrigen, char cadDestino)	Copia la cadena "cadOrigen" en la cadena "cadDestino"	char *s = "hola"; char *c = (char *) malloc (longitudCadena (s)+1); copiarCadenas(s, c); printf (" %s >>> %s", s, c);

char *copiarN
(char *cadOrigen, char *cadDestino, int n)
	Copia los primeros "n" caracteres de la cadena "cadOrigen" en la cadena "cadDestino"	char *s1 = "holamundo";
		w = 3;
		char *s2 = (char *) malloc (w+1);
		copiarN (s1, s2, 3);
		printf (" %s", s2);
		// imprime "hol"

char *copiarSub
(char *cadOrigen, char *cadDestino, int m, int n)
	Copia la subcadena dada por los caracteres entre la posición "n" y "m" de la cadena "cadOrigen" en la cadena "cadDestino"	char *s1 = "holamundo";
		m = 3; n=6;
		char *s2 = (char *) malloc (m-n+1);
		copiarN (s1, s2, m, n);
		printf (" %s", s2);
		// imprime "amun"

int compararCadenas
(char *cad1, char *cad2)
	Compara las cadenas "cad1" y "cad2" y devuelve:
	0 si cad1 = cad2
	< 0 si cad1 < cad2
	> 0 si cad1 > cad2.
	Tenga en cuenta que la comparación es lexicográfica
		i=compararCadenas ("MNP", "mnp");
		// resultado < 0
		i=compararCadenas ("abc", "abc");
		// resultado = 0
		i=compararCadenas ("xy", "abc");
		// resultado > 0

char * concatenarCadenas (char *cad1, char *cad2)	Concatena "cad2" a "cad1". El resultado queda en "cad1" y retorna esta cadena. (Utilize la funcion *realloc* para más memoria)	char *s1 = "hola"; char *s2 = "mundo" concatenarCadenas (s1, s2); printf (" %s", s1) // Imprime "holamundo"
char *asignarCadena (char *cad, char car)	Cambia todos los caracteres de "cad" al caracter	char *cad="abcd"; asignarCadena (cad,'x'); // cad es ahora xxxx
char *haciaMayusculas (char *cad)	Cambia los caracterés de la cadena "cad" a Mayusculas (si no son letras los deja igual).	char *cad="abcd$e"; haciaMayusculas (cad,'x'); // cad es ahora "ABCD$E"

int existeCaracter
(char *cad, caracter car)
	Retorna falso o verdadero si el caracter "car" existe en la cadena "cad"	char *cad="abcd$e";
		int e = existeCaracter (cad,'$');
		// e vale ahora 1

int posIniCaracter (char *cad, caracter car)	Retorna la posición en que se encuentra el primer carácter "car" desde el inicio. Si no lo encuentra, retorna -1	char *cad="abcd$eaw";
		int e = posIniCaracter (cad,'a');
		// e vale ahora 0

int posFinCaracter (char *cad, caracter car)	Retorna la posición en que se encuentra el primer caracter "car" desde el final. Si no lo encuentra, retorna -1.	char *cad="abcd$eaw";
		int e = posFinCaracter (cad,'a');
		// e vale ahora 6

2. Número Doblemágico

1. Transforme su programa de número mágico de tal forma que va a tener no uno SINO dos números mágicos. Ahora la validación si los adivina o no va a estar dada por las siguientes reglas:

a) Correcto si adivina uno de los dos números

b) Correcto si adivina la suma, resta, o multiplicación de los dos números.

c) De lo contrario debe informarle si el número que ingresó es:

1) Mayor o menor que el primer número mágico
2) Mayor o menor que el segundo número mágico
3) Está entre los dos números (numMagico1 < numeroUsurio < numMagico2).

2. Convierta el programa doble número mágico en una función que retorne el número de intentos que tomó para adivinar uno de los números (retorna -1 en caso contrario) y que reciba como parámetros el número máximo de intentos y un arreglo donde va a guardar la "historia" de los eventos, así:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Prototipo de la función
int numeroDobleMagico (int maxIntentos, int *historia);
// El llamado sería asi:
int main () {
  int nIntentos, i;
  int hist [100];
   nIntentos = numeroDobleMagico (10, hist)
  if (adivino > 0) { // Verdad que lo adivino
    for (i=0; i < nIntentos; i++) {
      printf ("%d", hist [i]);
    }
  else
    printf ("Fallaste...");
}

// Implementacion de la funcion
int numeroDobleMagico (int maxIntentos, int *historia) {
  ...
  ...
}
```

3. Preguntas

1. Why C is called a mid level programming language?
2. What are the features of C language? (Just 5 is good)
3. What is the difference between local variable and global variable in C?
4. What is array in C?
5. What is pointer in C?
6. Why are pointers dangerous in bad hands?
7. Why pointers are dangerous when handled incorrectly?
8. Can we access array using pointer in C language?
9. When is the "void" keyword used in a function?
10. What are the advantages of using pointers?
11. What is the difference between “break” and continue statements?
12. What is the difference between “Char a” and “Char a[1]”?
13. What is the difference between strings and arrays?
14. What are compilers?
15. How a negative integer is stored.?

16. What is a dangling pointer?
17. Where local variables are stored?
18. Can we assign a float variable to a long integer variable?
19. What it the return value of a relational operator if it returns any?
20. What is the default value of local and global variables?
21. Can a pointer access the array?
22. Why the string length is not n but n-1?
23. Functions must and should be declared. Comment on this.
24. What is the difference between C and Python (Comment on compilers and interpreters?
25. Which level is C language belonging to, and what it means?
26. What do you mean by high level, middle level and low level languages and give an example for each?
27. What is compiler?
28. What is the difference between assembler, compiler and interpreter?
29. Execution of a C program starts from which function?
30. Is C language case sensitive, what it means?
31. What is the difference between int, char, float and double data types?
32. What is the use of sizeof() function in C?
33. What are different types of modifiers in C?
34. What is local variable in C?
35. What is global variable in C?
36. What is the difference between single equal “=” and double equal “==” operators in C?
37. What is the difference between while and do-while loops in C?
38. What is “&” and “*” operators in C?
39. What will happen if break statement is not used in switch case in C?
40. What is the use of “#include” in C?
41. Is it necessary to declare the array size before using it, why?
42. Can array size be declared at run time?
43. What is meant by segmentation fault or memory fault in C?