Práctica 2 de Programación 1

Andrés Farías y Elvis Paez

Mayo, 2024

1) Introducción

El problema a resolver consiste en un traductor de mensajes. Para el cual usted debe programar y usar un conjunto de funciones y procedimientos que permitan resolver pequeños problemas para llevar a cabo las operaciones necesarias.

Para tener el Traductor de Mensajes se requiere de una cadena inicial que estará expresada en un formato binario y será convertida a un formato alfanumérico. El objetivo será hacer las respectivas operaciones para poder traducir el texto enviado al programa.

Un ejemplo de esto sería:
-Dada la cadena 01001000011011110110110001100001

En donde:

01001000 es ’H’

01101111 es ’o’

01101100 es ’l’

01100001 es ’a’

La salida traducida debería ser ”Hola”.

2) Práctica

Para la realización de este trabajo, se deberá hacer uso de lo visto en clases, sin añadir ninguna otra biblioteca adicional a “stdio.h” y sin añadir contenido adicional a lo explicado en el curso hasta los momentos, abarcando el manejo de los tipos de datos int, char y registros, así como sus respectivos apuntadores. Sumado a esto se utilizará su conocimiento sobre funciones, procedimientos, estructuras de repetición y de decisión.

Casos de prueba:

Input: 01001000011011110110110001100001 Output: Hola Input: 010100000111001000110001010101010110110001100001 Output: Pr1Ula
Input: 010001010111100001100001011011010110010101101110 Output: Examen

Estos casos de prueba le servirán como una ayuda para verificar que su programa funcione bien, pero los evaluados serán distintos a los aquí expresados.

3) Cosas a considerar antes de empezar

Es necesario que usted como desarrollador comprenda el funcionamiento de dos aspectos fundamentales dentro de la programaci´on:

a) Representación en código ASCII: Todos los caracteres dentro del idioma inglés pueden ser representados mediante un valor dentro del código ASCII, es por esto entonces que los caracteres pueden considerarse como números dentro de este código.

Ejemplo:

Caracter: A Código en ASCII: 65
Caracter: W C´código en ASCII: 87
Caracter: a Código en ASCII: 97
Caracter: j Código en ASCII: 106
Caracter: 1 Código en ASCII: 49

b) Código binario: El código binario corresponde a codificación que permite escribir números de base decimal (0,1,2,3,4....) en secuencias de 8 dígitos en base binaria (00000000,00000001,00000010,00000011....), donde cada dígito del número binario es llamado bit.

0 1 0 0 0 0 0 1

c) Lectura de números binarios: Los números binarios a diferencia de los números de base decimal deben leerse de derecha a izquierda, es decir, empezando desde el final, donde cada bit posee una representación de una potencia de 2:

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Donde la transformación de este número a binario es la suma de las potencias cuyos bits sean 1, por ejemplo:

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$
0 1 0 0 0 0 0 1

Este sería el n´umero $2^6 + 2^0 = 64 + 1 = 65$

d) Representación de caracteres en formato binario: Todos los números de la tabla ASCII normal pueden ser representados mediante un código binario de ocho bits denominado ”byte”, es por esto que la tabla ASCII posee su propia interpretación en binario. De esta manera podemos comprender que cada caracter puede ser llevado a su representación en el código ASCII y de allí convertirse en su símil binario.

Ejm:

Caracter: A C´odigo en ASCII: 65 Representación en binario: 01000001

4) Enunciado

La agencia espacial NASA le requiere a usted para la creación de un programa que les permita enviar instrucciones a sus astronautas a bordo de una nave espacial. Dadas las complicaciones para la transferencia de grandes paquetes de información, la agencia espacial ha determinado que la mejor manera de enviar datos es en forma de pulsos interpretados como señales binarias. A pesar de esto los astronautas requieren poder visualizar el mensaje en un formato de texto alfanumérico, para ello se le solicitan las siguientes rutinas.

5) Rutinas

1. Calentamiento:

a) **int potentiation(int, int)**: La cual recibe un entero base y un entero exponente, cumple con la definición matemática de la potenciación, deberá retornar el valor de elevar la base a la potencia.

b) **char int_to_char(int)**: La cual recibe un dato de tipo entero y deberá retornar el caracter representado en la tabla ASCII.

c) **void print_string(char*)**: Recibe un apuntador a caracter. El procedimiento deberá imprimir su contenido por el terminal. Este subprograma debe realizarse en una sola instrucci´on de código

2. Subprogramas:

a) **int binary_to_int(char*)**: Esta función recibe como parámetro un apuntador al arreglo de caracteres, el arreglo contendrá un “byte”, es decir 8 bits, los cuales deberán ser convertidos y retornados en su interpretación en formato decimal. Para la realización de esta rutina se recomienda utilizar la función ***pow(int, int)*** previamente descrita, además asuma que la cadena contendrá siempre la cantidad exacta de 8 caracteres.

b) **char* add_character(char*, unsigned int*, char)**: Esta función recibe como parámetros un apuntador a caracter que es la cadena a la cual se le va a agregar un nuevo caracter, un apuntador a entero que refencia al tamaño actual de dicha cadena y finalmente un caracter, su objetivo es retornar la cadena recibida a la cual se le habría

adjuntado al final el caracter recibido. **ATENCIÓN**: Se debe tener en cuenta que, al agregar un caracter a una cadena se debe actualizar su tamaño.

c) **char* extract_string(char*, int, int, char*)**: Esta función recibe un apuntador a una cadena de caracteres de la cual se extraerá la subcadena, dos datos de tipo entero, los cuales expresan el punto inicial y final desde donde se extraerá la subcadena de la original (estos puntos inclusive) y finalmente otro apuntador a caracter el cual servirá para almacenar los datos de la nueva cadena.

d) **int main():** En la función principal se debe leer primeramente una cadena de caracteres binarios que representa 2 bytes (16 bits) el cual contiene el número de bits que serán leídos a continuación, posteriormente se debe leer esa cantidad de bits por pantalla y hacer un llamado a la función ***translate_message(char*, int, Logger)*** enviando la cadena binaria leída y la cantidad de bytes que contiene la misma. Asuma que la cantidad de bits siempre representa un número exacto de bytes.

6) Ayuda

Para ayudarte a probar tu solución, en la plantilla se incluye una serie de elementos que pueden amenizar tu proceso creativo:

• **void translate_message(char*, int, Logger):** Este procedimiento contendrá una implementación que hará los llamados correspondientes para traducir el mensaje por ti, ahora solo debes preocuparte de cumplir correctamente con las rutinas solicitadas. El funcionamiento del mismo es sencillo, este recibe un apuntador a una cadena en formato binario la cual será traducida, así como también un entero que guardará la cantidad de “bits” contenidos en la misma.

Entrada:

0000000000111000	| 56
10000011101100110110010000011011111101011100001	| All ok!

7) Recomendaciones

1. Lea enteramente este enunciado antes de proceder al diseño e implementación. Asegúrese de comprender bien su diseño. Haga un boceto de su programa y como es esperado que funcione. Por cada rutina, plantee qué es lo que va a hacer, cómo va a resolver el problema. Esta es una situación que amerita de diseño y desarrollo.

3. No se debe modificar la plantilla que se provee, los prototipos facilitados deben estar tal cual acá planteados, tampoco debe agregar parámetros adicionales, no debe crear más funciones o procedimientos que los acá descritos, al añadir nuevas variables

debe asignarles nombres que ayuden a comprender el funcionamiento del código.

4. Escribe tu código como si fuese un poema para esa persona que tanto te atrae, de fácil comprensión, con una elegancia y estilo apropiado.