**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ZEPEDA FLORES ALEJANDRO DE JESÚS**

**NO. BOLETA 2016601853**

**TEORÍA COMPUTACIONAL**

**PROF. LUZ MARÍA SANCHÉZ GARCÍA**

**2 DE MARZO DE 2018**

**INTRODUCCIÓN**

La práctica de las operaciones de cadenas definidas sobre un alfabeto V sirve para mostrar la forma en cómo se realizan las operaciones; este proceso no es complicado, ya que se trabaja mediante cadenas ingresadas por el usuario y a partir de eso, aplicamos las operaciones en las mismas.

Además, mediante los resultados obtenidos en este reporte podremos analizar y compararlos; aunque el problema es el mismo, la implementación es diferente. Estos, servirán como base para detectar errores (si existe algún error) o encontrar mejoras para lograr una implementación más adecuada.

**PLANTEAMIENTO DEL PROBLEMA**

Realizar las operaciones de cadenas sobre un alfabeto V={a-z}. La implementación se desarrollará en lenguaje C, donde el usuario podrá seleccionar la operación que desee y este solicitará las cadenas pertinentes para llevar a cabo dicha operación.

Se deben tomar en cuenta las propiedades de las operaciones con cadenas; de no hacerlo, el resultado obtenido será totalmente incorrecto y el propósito principal de esta práctica no será cumplido.

**DIAGRAMA**

FIN

Elaborar reporte.

NO

SI

Corregir errores y realizar pruebas.

Existen errores

Realizar las pruebas necesarias para checar la implementación.

Codificar la implementación.

Hacer un análisis de cómo implementar la idea del primer proceso.

Seleccionar el lenguaje de programación donde se tenga mayor habilidad para desarrollar la solución.

Plantear la solución al problema de implementar las operaciones de cadenas en un lenguaje de programación.

INICIO

**IMPLEMENTACIÓN DE LA SOLUCIÓN**

//Zepeda Flores Alejandro de Jesús - 2CM4 22/Febrero/2018

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Practica1.h"

int main(int argc, char \*argv[]){

int x = 0, opt = 0, repeat = 0, exponente = 0;

char cadena1[100]; char cadena2[100];

do{ //Ciclo do while para ofrecer al usuario la opción de realizar otras operaciones sin la necesidad de cerrar el programa

printf("Practica 1 - Alejandro Zepeda Flores\n\n"); //Interfaz para hacer más accesible el uso del programa

printf("1-. Palindromo\n2-. Longitud\n3-. Concatenacion\n4-. Potencias\n5-. Inverso\n6-. Prefijos\n\n");

printf("Opcion: "); scanf("%d", &opt);

switch(opt){

case 1: //Caso 1: Cadena palindromo

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

printf("\nCadena original: %s\n",cadena1);

printf("Cadena invertida: %s",invertir(cadena1));

if(palindromo(cadena1)) printf("\n\nLa cadena es un palindromo\n\n");

else printf("\n\nLa cadena no es un palindromo\n\n");

repeat = 1; system("pause");

system("cls");

break;

case 2: //Caso 2: Longitud de la cadena

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

printf("\nLongitud de la cadena %s: %d\n\n",cadena1,longitud(cadena1));

repeat = 1; system("pause");

system("cls");

break;

case 3: //Caso 3: Concatenación de dos cadenas

system("cls");

fflush(stdin); printf("Cadena 1: "); gets(cadena1);

fflush(stdin); printf("Cadena 2: "); gets(cadena2);

printf("\nCadena concatenada: %s\n\n",concatenar(cadena1,cadena2));

repeat = 1; system("pause");

system("cls");

break;

case 4: //Caso 4: Potencia de una cadena

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

printf("Potencia: "); scanf("%d", &exponente);

if(potencia(cadena1,exponente) == NULL)

printf("\nNueva cadena %c\n\n",156);

else

printf("\nNueva cadena: %s\n\n",potencia(cadena1,exponente));

repeat = 1; system("pause");

system("cls");

break;

case 5: //Caso 5: Invertir cadena

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

printf("\nCadena invertida: %s\n\n",invertir(cadena1));

repeat = 1; system("pause");

system("cls");

break;

case 6: //Caso 6: prefijos de una cadena

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

prefijos(cadena1);

repeat = 1; system("pause");

system("cls");

break;

case 7: //Caso 7: posfijos de una cadena

system("cls"); fflush(stdin);

printf("Cadena: "); gets(cadena1);

sufijos(cadena1);

repeat = 1; system("pause");

system("cls");

break;

case 8: //Caso 8 subacadenas de una cadena

system("cls");

fflush(stdin); printf("Cadena: "); gets(cadena1);

buscar\_subcadena(cadena1); printf("\n\n");

repeat = 1; system("pause");

system("cls");

break;

case 9: //Caso 9: salir del programa

repeat = 0;

break;

default:

repeat = 1;

system("cls");

}

}while(repeat == 1);

return 0;

}

int palindromo(char \* cadena){

int i = 0, j = 0, palindromo = 1;

j = longitud(cadena)-1; //Calcular la longitud de la cadena

for(i=0; i<longitud(cadena)/2 ; i++,j--){ //Ciclo for para recorrer los extremos de la cadena

if(\*(cadena+i) != \*(cadena+j)){ //SI un caracter no coincide, terminar el ciclo

palindromo = 0; break; //Terminamos el ciclo y actualizamos palindromo

}

}

if(palindromo) return 1; //Si palindromo = 1, significa que la cadena es palindromo

else return 0;

}

int longitud(char \* cadena){

return strlen(cadena); //Retorna la longitud de una cadena

}

char \* concatenar(char \* cadena1, char \* cadena2){

char \* auxiliar = NULL;

int x = 0, y = 0, tam1 = 0, tam2 = 0;

tam1 = longitud(cadena1); //Calcular tamaño de cadena1

tam2 = longitud(cadena2); //Calcular tamano de cadena 2

auxiliar = (char \*)malloc(sizeof(char)\*(tam1+tam2)); //Solicitar el espacio de memoria para la suma de los tamaños

while(x<tam1){ auxiliar[x] = cadena1[x]; x++; } //Asignar a la nueva cadena la cadena 1

while(y<=tam2){ auxiliar[x] = cadena2[y]; x++; y++; } //Asignar a la nueva cadena la cadena 2

auxiliar[x] = '\0'; //Asignar el fin de cadena

return auxiliar;

}

char \* potencia(char \* cadena, int exponente){

int i = 0, j = 0, tam = 0, tam2 = 0;

char \* auxiliar = NULL; char \* auxiliar2 = NULL;

if(exponente == 0) return NULL; //Si exponente = 0 retorna epsilon

else if(exponente == 1) return cadena; //Si exponente = 1 retorna la misma cadena

else if(exponente == -1) return invertir(cadena); //Si exponente = -1 retorna la cadena inversa

if(exponente>1){ //Mientras exponente > 1

tam = longitud(cadena); //Obtener longitud de cadena

auxiliar = (char \*)malloc((sizeof(char))\*((tam\*exponente)+1)); //Solicitar los caracteres de la nueva cadena

for(i=0; i<(tam\*exponente) ;i++){ //Ciclo for para recorrer la longitud de la nueva cadena

auxiliar[i] = cadena[j]; j++; //Asignar a la nueva cadena la original

if(j==tam) j = 0; //Si j == tam, significa que recorrimos toda la cadena original

}

auxiliar[i] = '\0';//Asignar el fin de cadena

return auxiliar;

}

if(exponente<-1){

auxiliar2 = invertir(cadena); //Invertir la cadena

tam = longitud(auxiliar2); //Obtener longitud de cadena

auxiliar = (char \*)malloc((sizeof(char))\*((tam\*(exponente\*(-1))+1))); //Solicitar los caracteres de la nueva cadena

for(i=0; i<(tam\*exponente) ;i++){ //Ciclo for para recorrer la longitud de la nueva cadena

auxiliar[i] = auxiliar2[j]; j++; //Asignar a la nueva cadena la original

if(j==tam) j = 0; //Si j == tam, significa que recorrimos toda la cadena original

}

auxiliar[i] = '\0'; //Asignar el fin de cadena

return auxiliar;

}

}

char \* invertir(char \* cadena){

int i = 0, j = 0, tam = 0;

char \* auxiliar = NULL;

tam = longitud(cadena); //Obtener la longitud de la cadena

auxiliar = (char \*)malloc(sizeof(char)\*tam); //Solicitar la memoria para la nueva cadena

for(i=tam-1; i>=0 ; i--){ //Ciclo for inverso para recorrer la cadena

auxiliar[j] = cadena[i]; j++; //Asignar la nueva cadena la original invertida

}

auxiliar[j] = '\0'; //Asignar el fin de cadena

return auxiliar;

}

void prefijos(char \* cadena){

int i = 0, j = 0, l = 0, tam = 0;

char auxiliar[100][100];

tam = longitud(cadena); //Obtener la longitud de la cadena

for(i=0; i<=tam ;i++) //Asignar a todas las posiciones de la primer columana el caracter vacío

auxiliar[i][0] = 156;

for(i=0; i<=tam ;i++){ //Ciclo for que recorre columnas

for(j=1; j<=tam ;j++,l++) //Ciclo for que recorre filas

auxiliar[i][j] = cadena[l]; l=0; //Asignar la cadena original al nuevo original

}

for(i=0; i<tam+1 ;i++){ //Primer ciclo for para imprimir

if(i<9) printf("Prefijo 0%d: ",i+1); //Imprimir prefijos

else printf("Prefijo %d: ",i+1); //Imprimir prefijos

for(j=0; j<tam+1 ;j++){ //Segundo ciclo for para imprimir

if(j<=i)

printf(" %c", auxiliar[i][j]); //Imprimir los caracteres correspondientes

if(j==i)

printf("\n");

}

}

}

void sufijos(char \* cadena){

int i = 0, j = 0, l = 0, tam = 0;

char auxiliar[100][100]; char nueva[100];

strcpy(nueva,invertir(cadena)); //Invertir la cadena original

tam = longitud(nueva); //Obtener la longitud de la cadena

for(i=0; i<=tam ;i++) //Asignar a la primera posición de la primer columna

auxiliar[i][0] = 156;

for(i=0; i<=tam ;i++){ //Primer ciclo for para asignar

for(j=1; j<=tam ;j++,l++) //Segundo ciclo for para asignar

auxiliar[i][j] = nueva[l]; l=0; //Asignarl la cadena al nuevo arreglo

}

for(i=0; i<tam+1 ;i++){ //Primer ciclo for para imprimir

if(i<9) printf("Posfijo 0%d: ",i+1); //Imprimir posfijos

else printf("Posfijo %d: ",i+1); //Imprimir posfijos

for(j=tam+1; j>=0 ;j--){ //Segundo ciclo for para imprimir

printf(" %c ",auxiliar[i][j-1-i]); //Imprimir los caracteres correspondientes

}

printf("\n");

}

}

void buscar\_subcadena(char \* cadena1){

int l = 0, tam = 0, i = 0, p = 0, k = 0, j = 0, m = 0;

p = tam = longitud(cadena1); //Obtener la longitud de la cadena

for (l=0; l<tam; l++){ //Primer ciclo for para recorrerla

for (i=0; i<p; i++){ //Segundo ciclo for para recorrerla

k = k + 1; printf("\n");

for (j = m; j < k ;j++) //Ciclo for para imprimir

printf("%c",cadena1[j]); //Imprimir los caracteres correspondientes

}

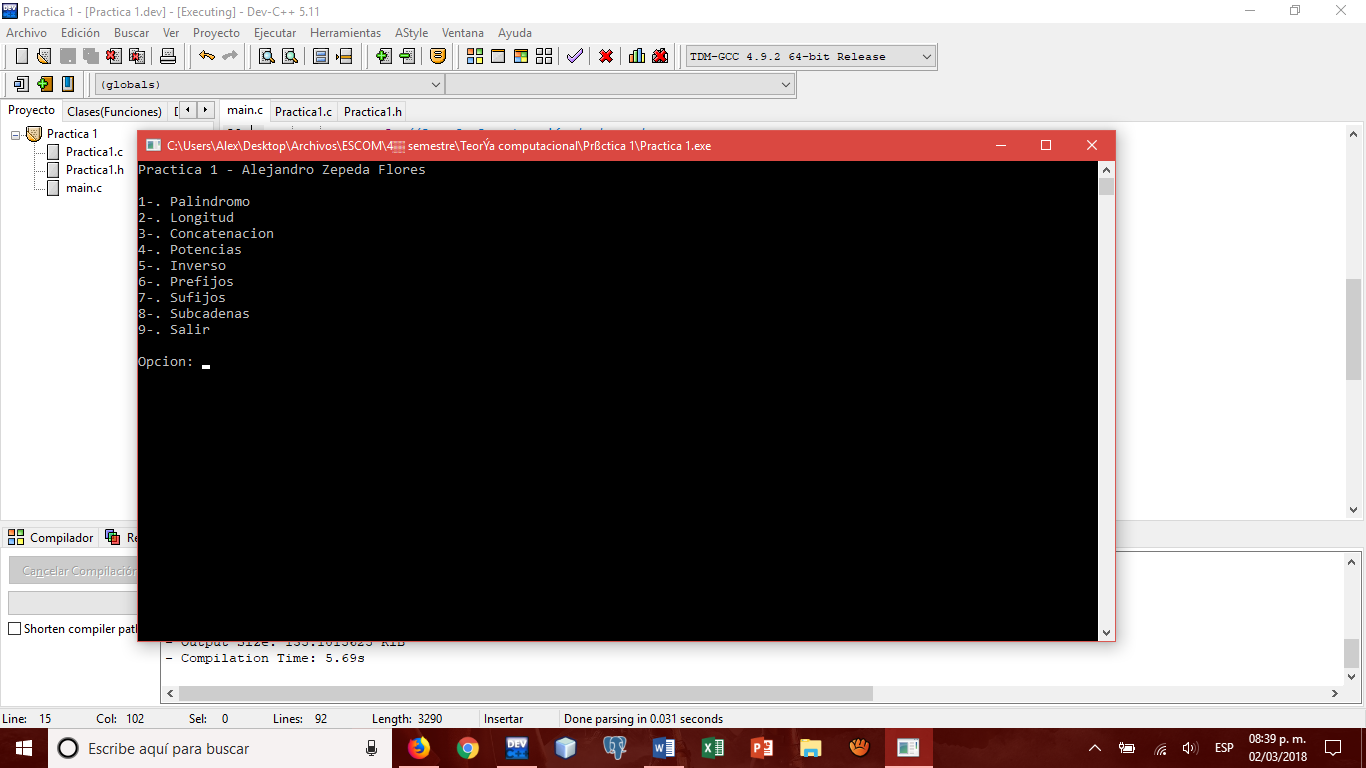
m = m + 1; p = p - 1; //Aumento de contadores

k = m; printf("\n");

}

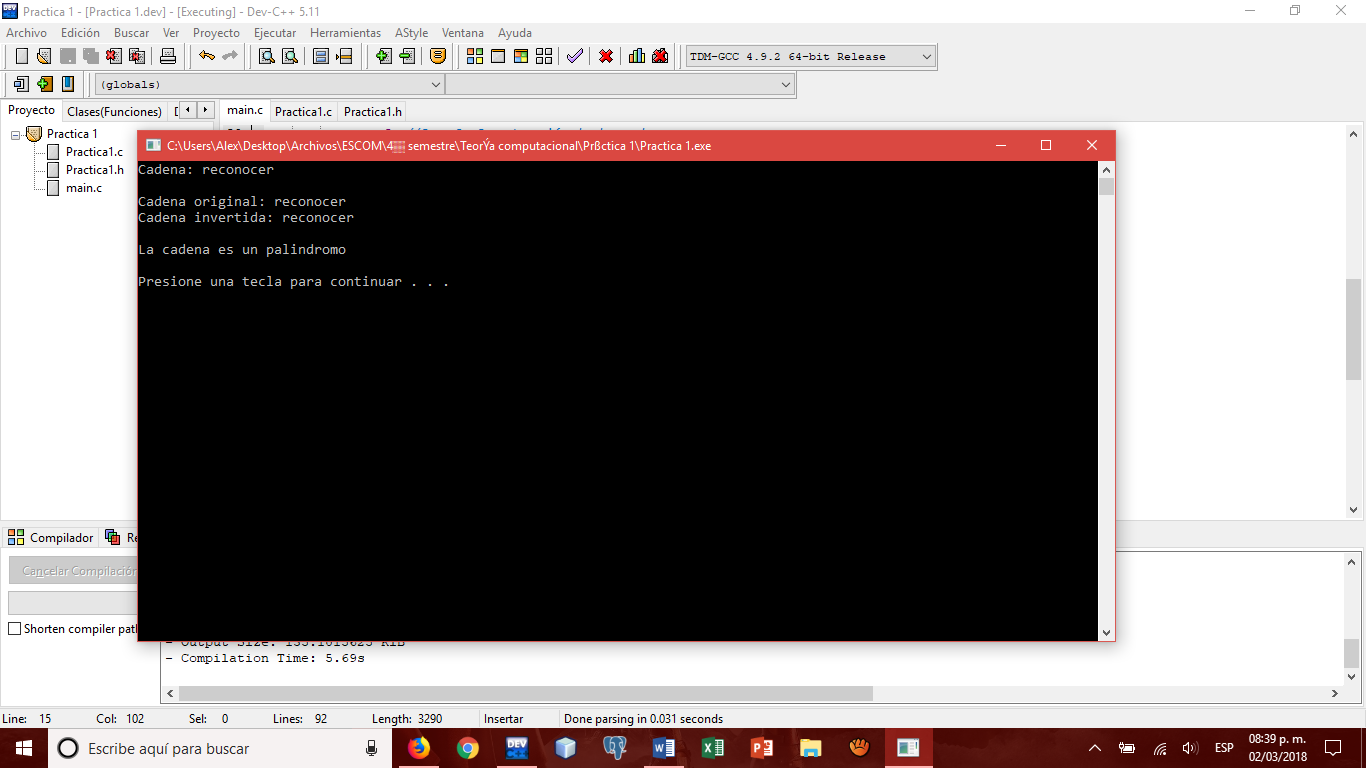
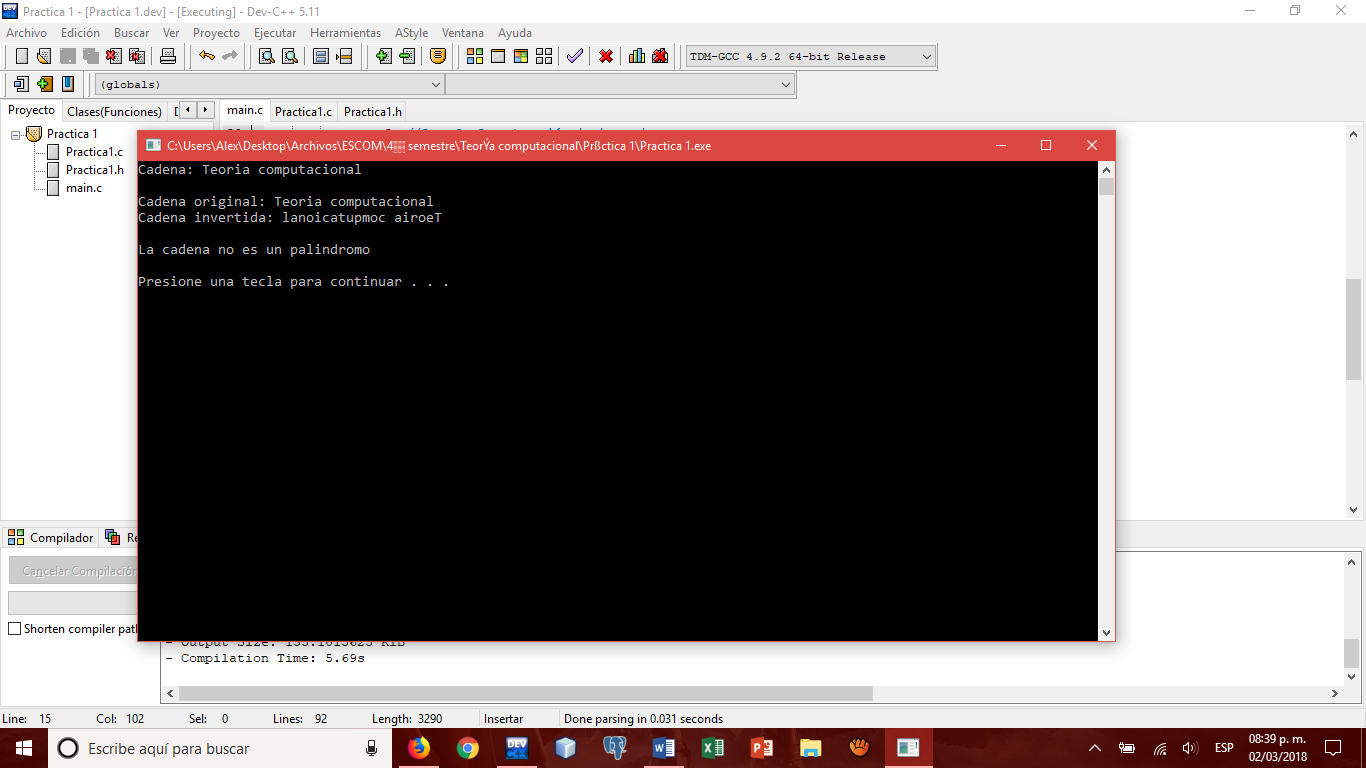
}

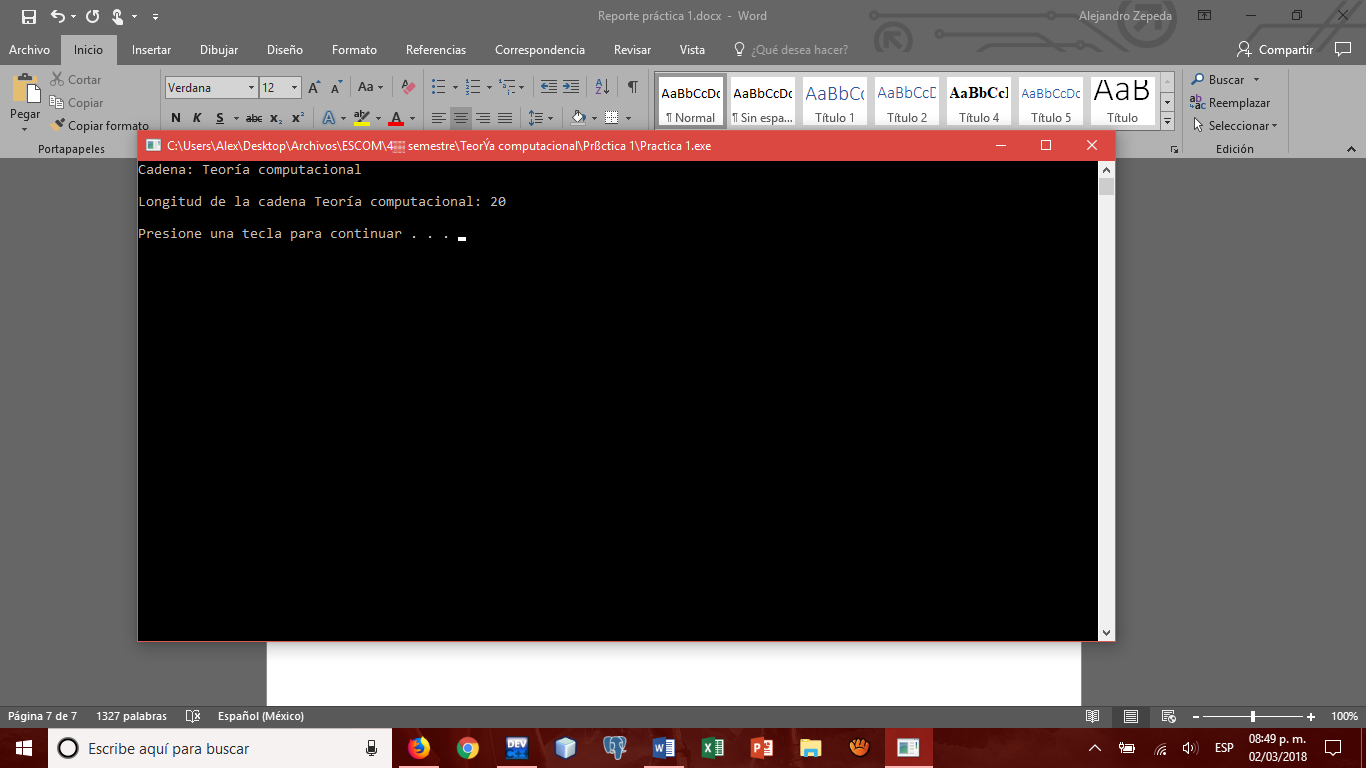
**FUNCIONAMIENTO**

****

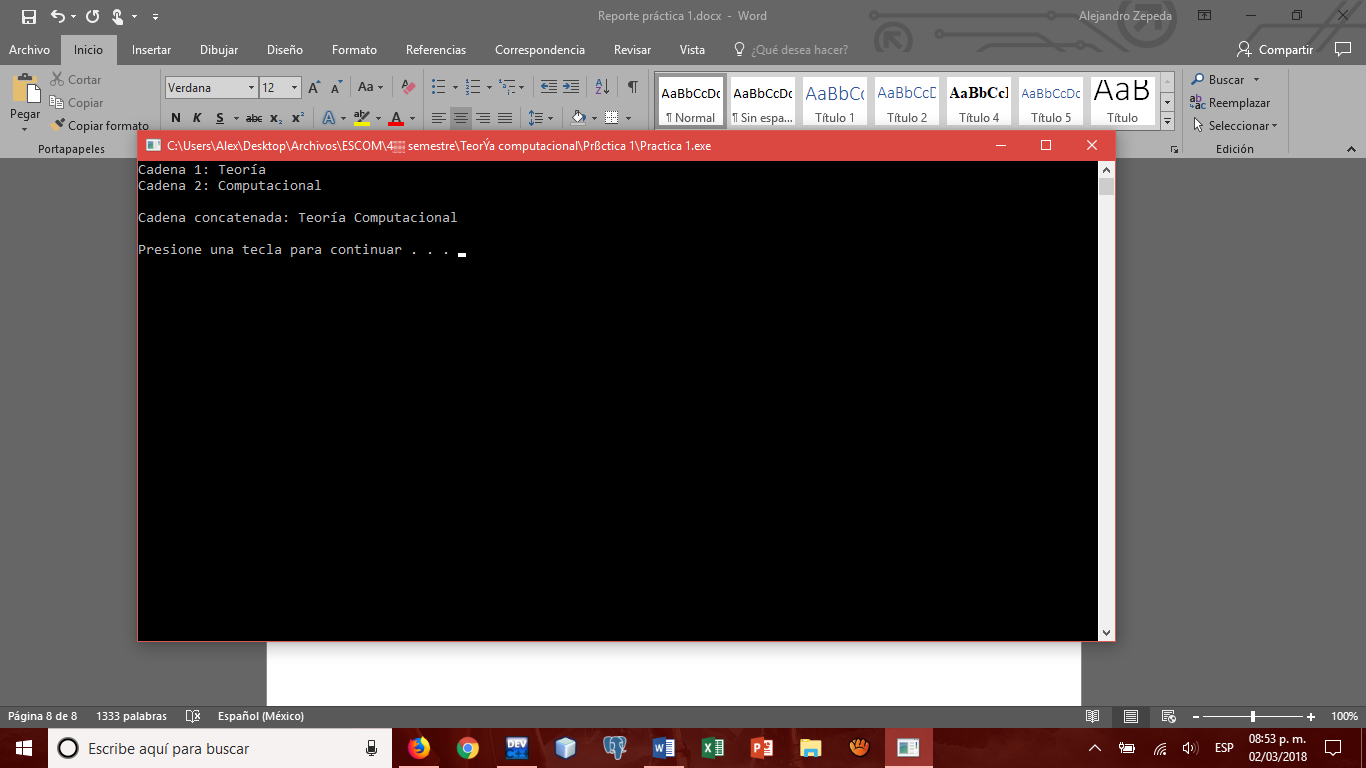
Interfaz de usuario

Primera operación – Cadena palíndroma

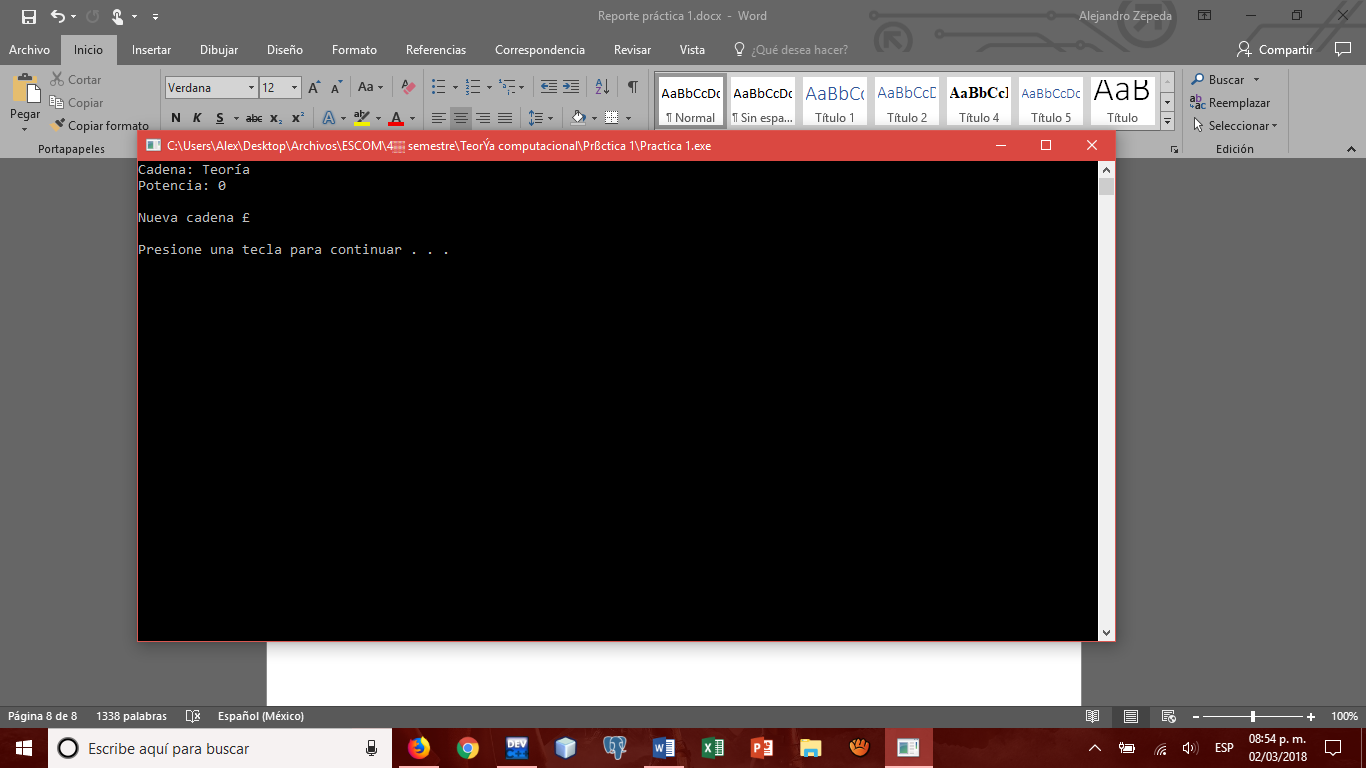
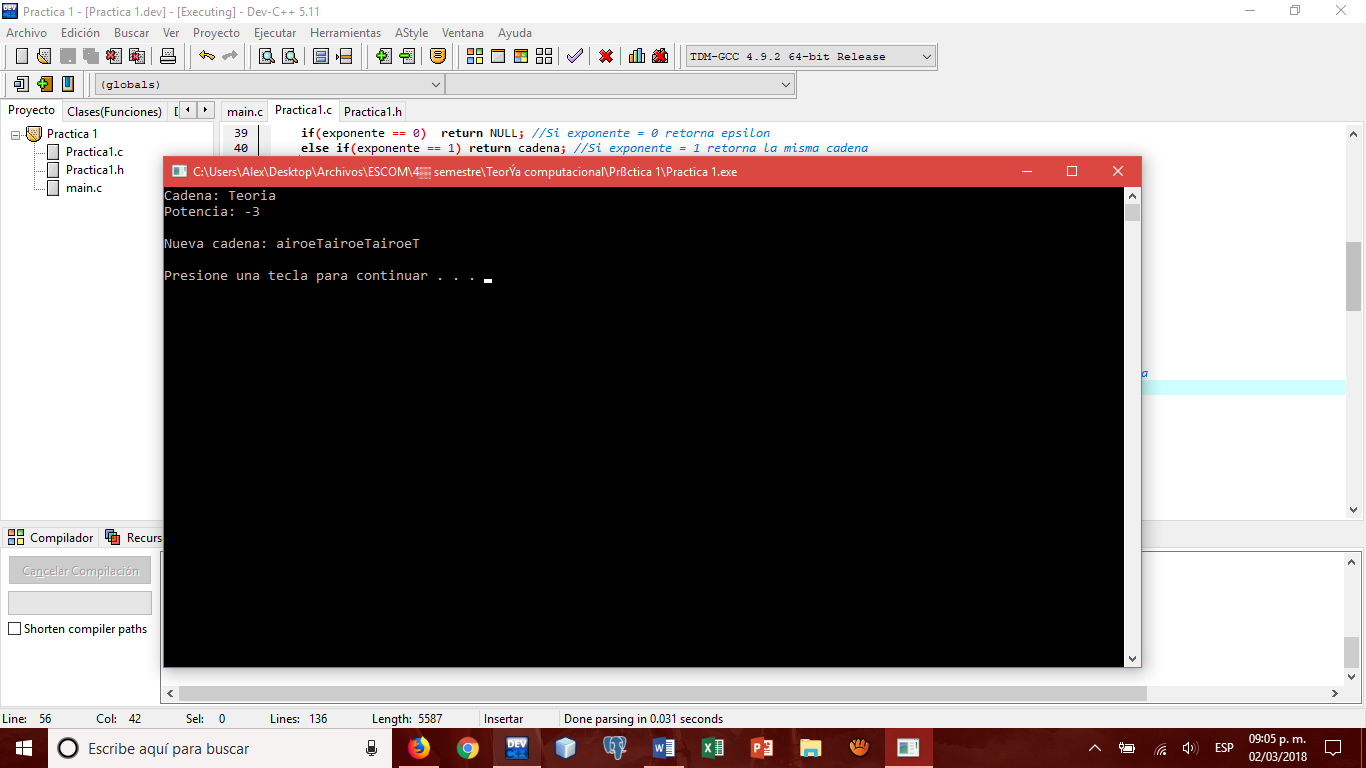
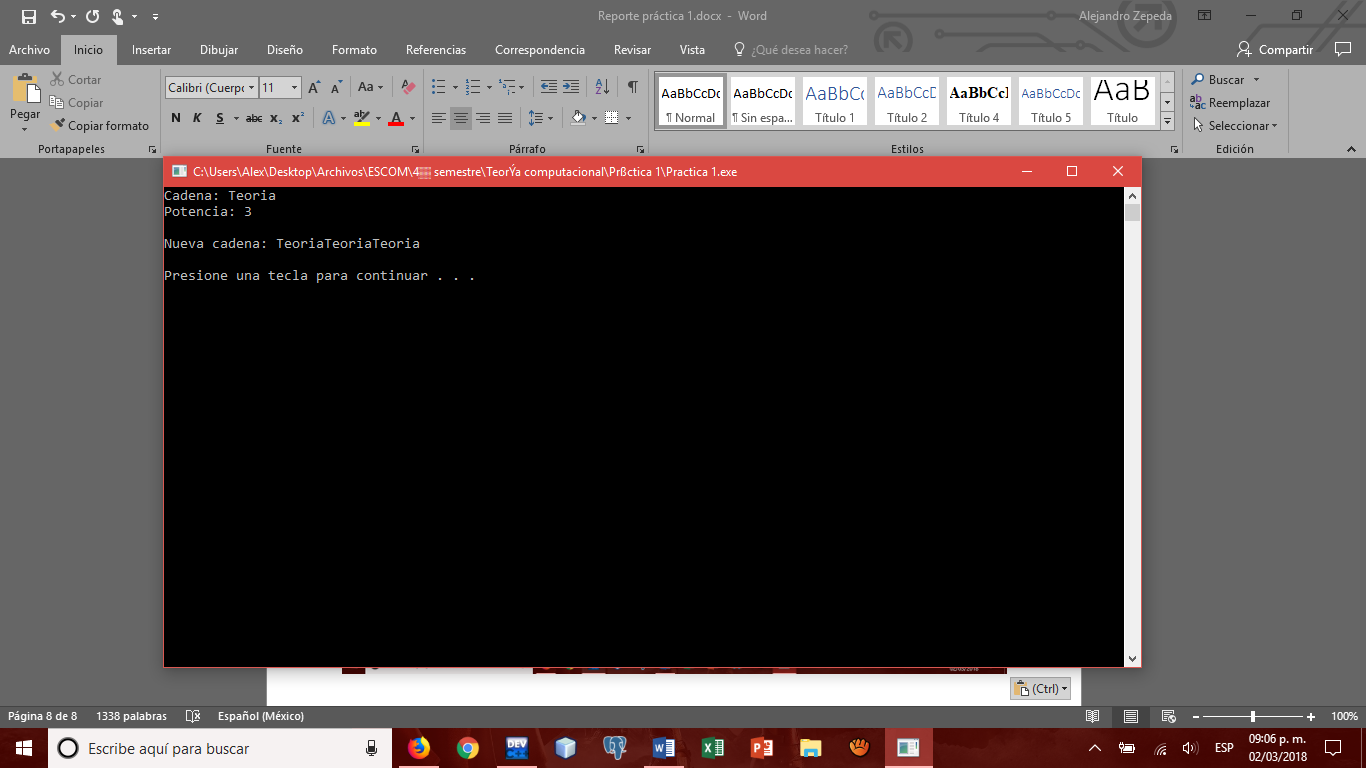


Segunda operación – Longitud de una cadena

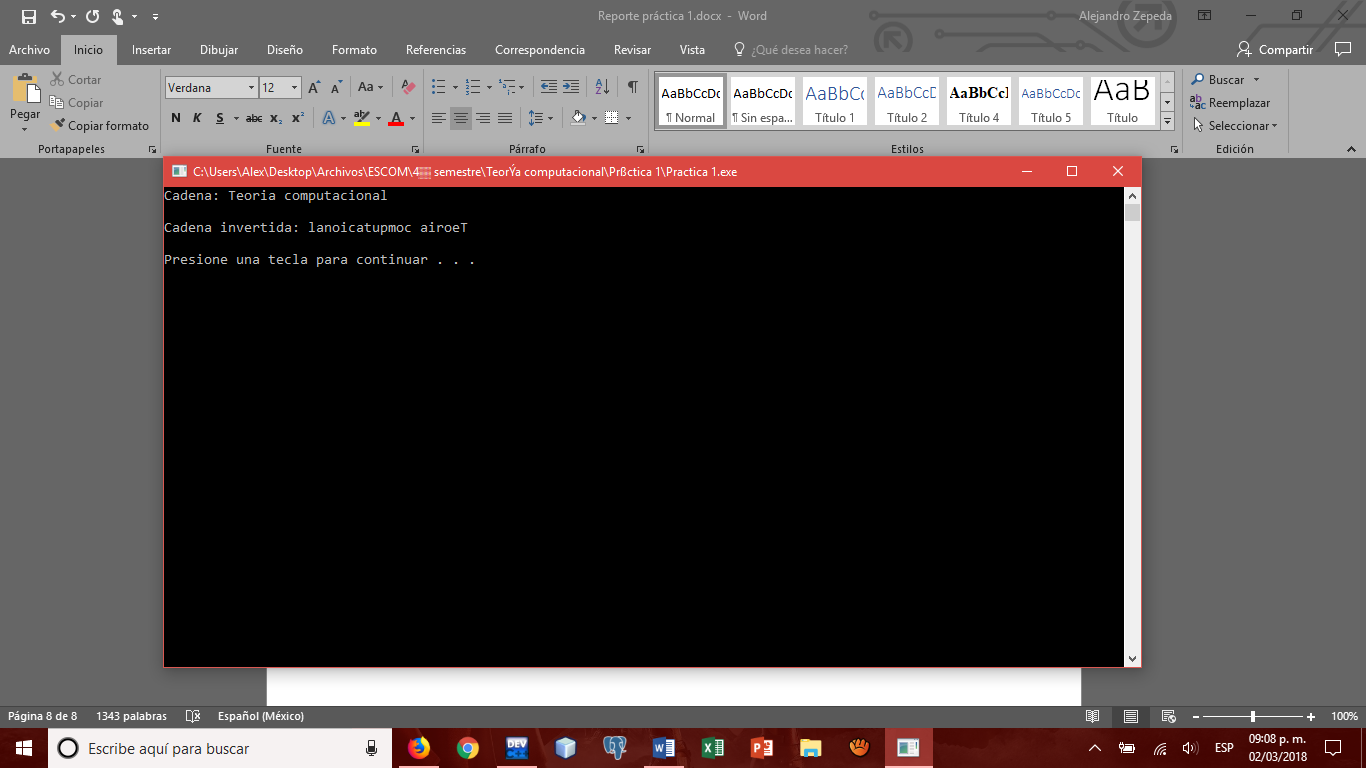
Tercera operación – Concatenación de cadenas



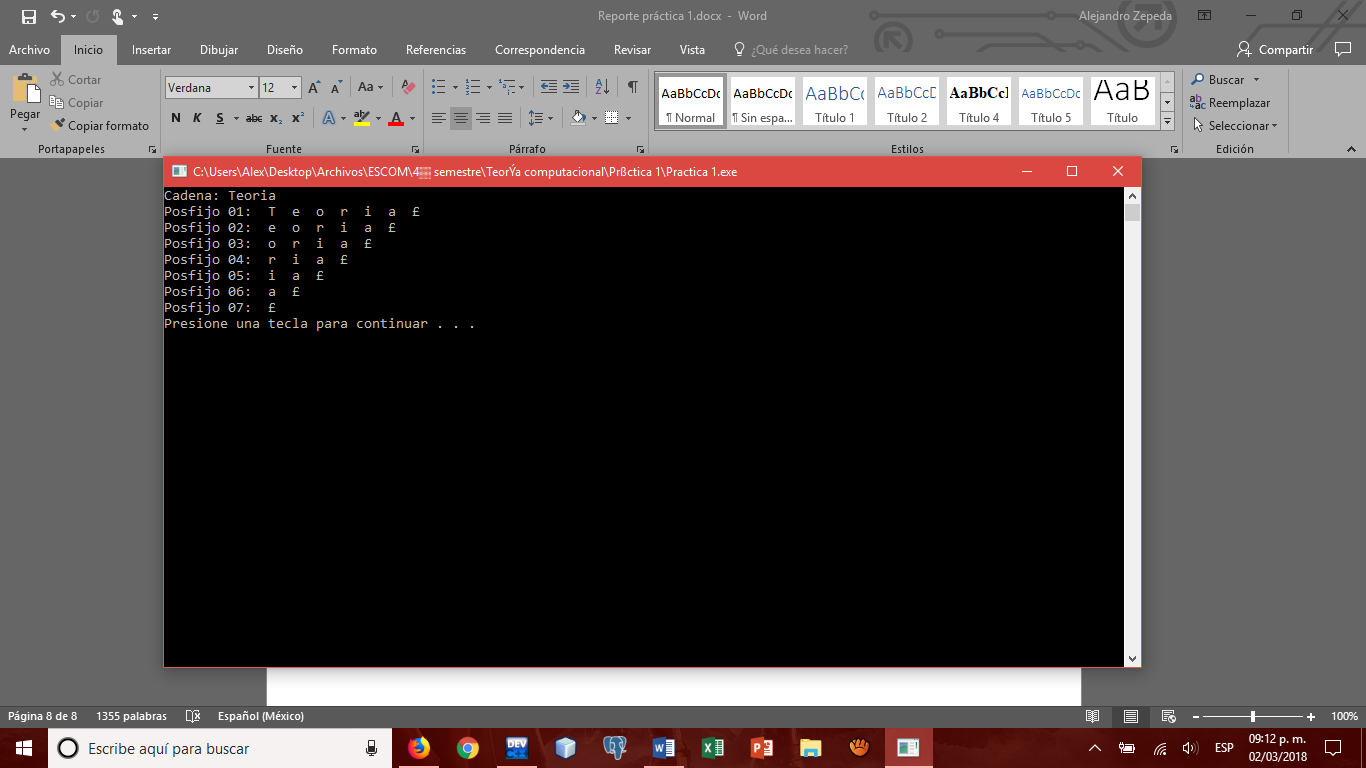
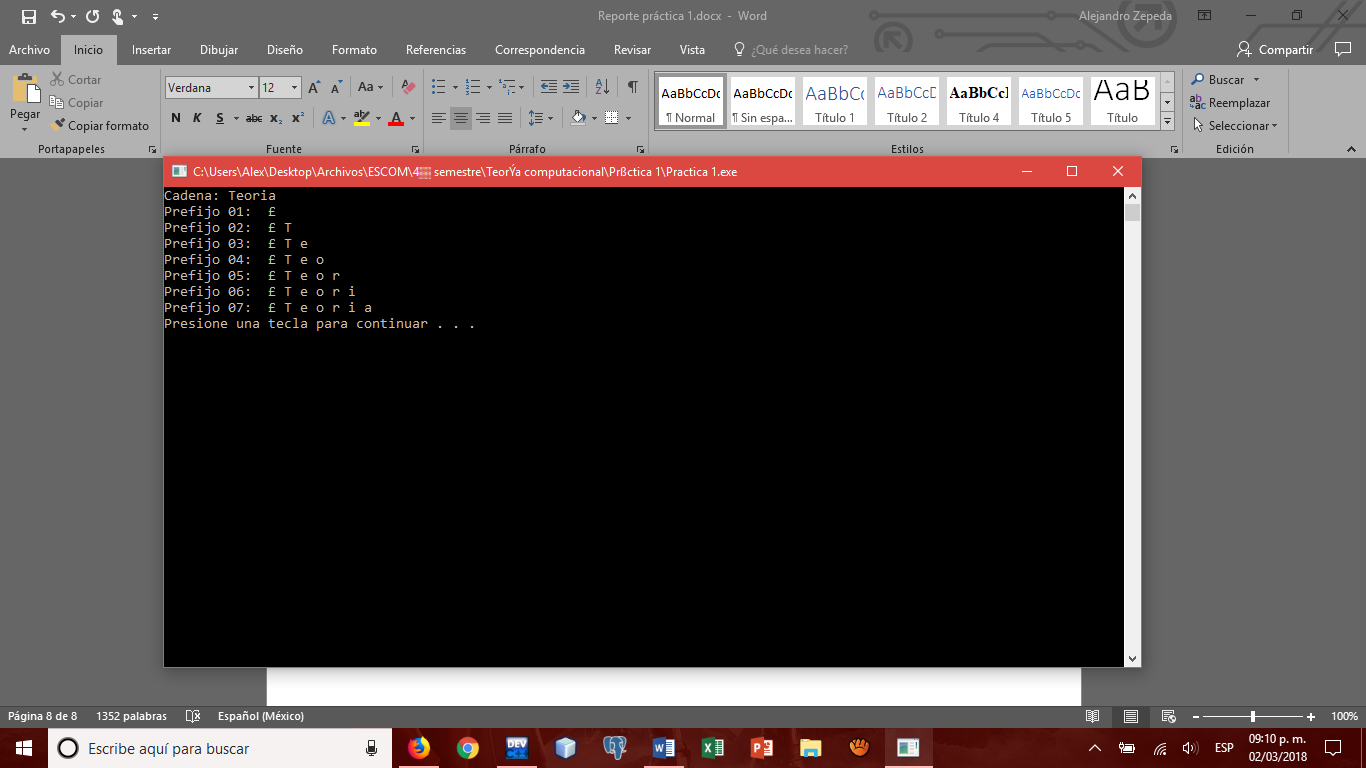
Cuarta operación – Potencia de cadenas

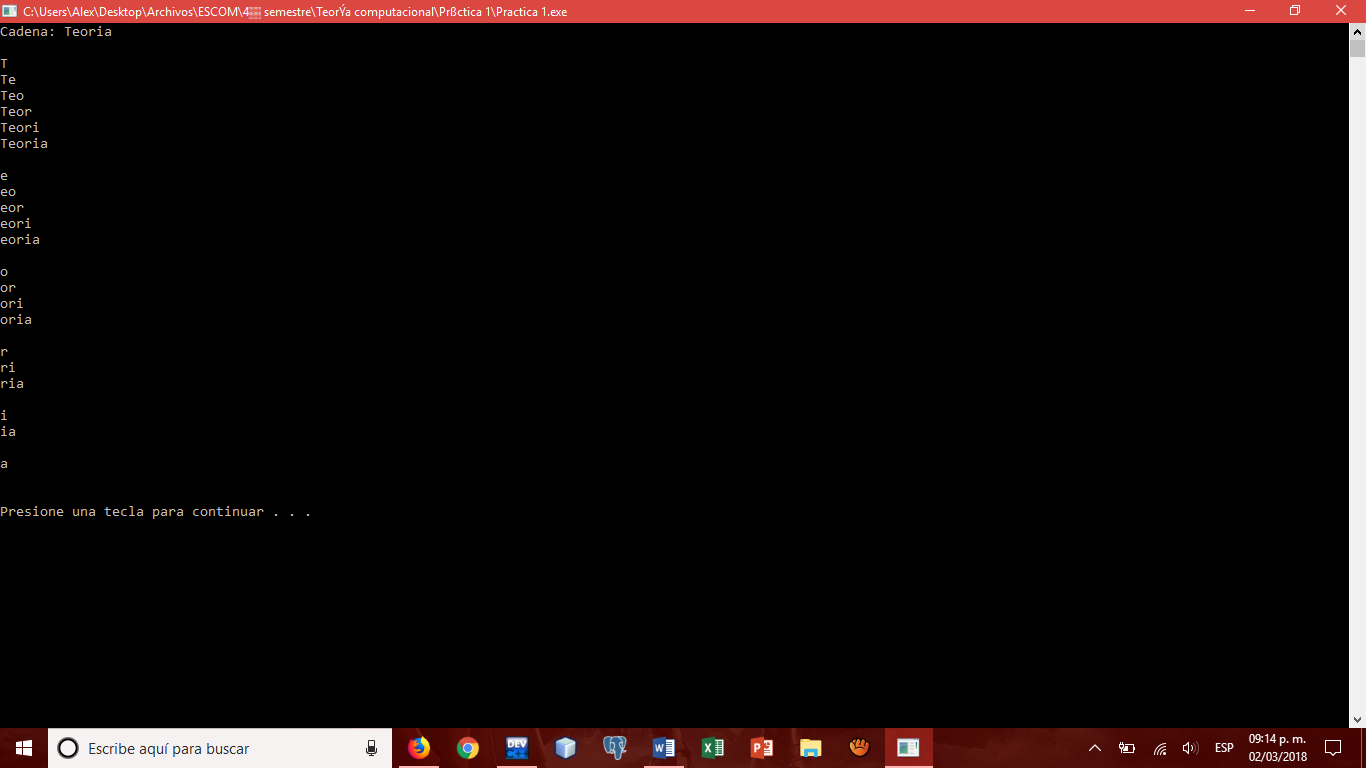


Quinta operación – Inverso de una cadena



Sexta operación – Prefijos Séptima operación – Posfijos



Octava operación – Subcadenas

**CONCLUSIÓN**

Un alfabeto es un conjunto finito no vacío de símbolos. A partir de un alfabeto podemos formar cadenas, que son una secuencia finita de símbolos pertenecientes a un alfabeto. Ahora, a partir de cadenas podemos formar lenguajes que son cualquier conjunto de cadenas formadas con símbolos de un cierto alfabeto.

Esto tiene una gran importancia, desde mi punto de vista, en la elaboración e identificación de lenguajes, tipos de expresiones y de autómatas. Además, aclaró la forma en cómo se realizan las operaciones de cadenas; en un principio era complicado, pero como para programar necesitas primero entender lógicamente como se realiza el proceso, facilitó la parte lógica de las operaciones.

**BIBLIOGRAFÍA**

* Dean Kelly. (1995). Teoría de Autómatas y Lenguajes Formales. España: Prentice Hall.
* Ramón Brena. (2003). Autómatas y Lenguajes. Marzo 03, 2018, de Instituto Tecnológico y de Estudios Superiores de Monterrey Sitio web: http://fcbinueva.unillanos.edu.co/docus/Automatas%20Y%20LenguajeL.pdf
* Holger Billhardt. (2008). 1 Teoría de Autómatas y Lenguajes Formales. Marzo 02, 2018, de Universidad Rey Juan Carlos Sitio web: http://www.ia.urjc.es/grupo/docencia/automatas\_itis/apuntes/capitulo%201.ppt.pdf