INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ZEPEDA FLORES ALEJANDRO DE JESÚS

NO. BOLETA 2016601853

TEORÍA COMPUTACIONAL

PROF. LUZ MARÍA SANCHÉZ GARCÍA

18 DE MARZO DE 2018

INTRODUCCIÓN

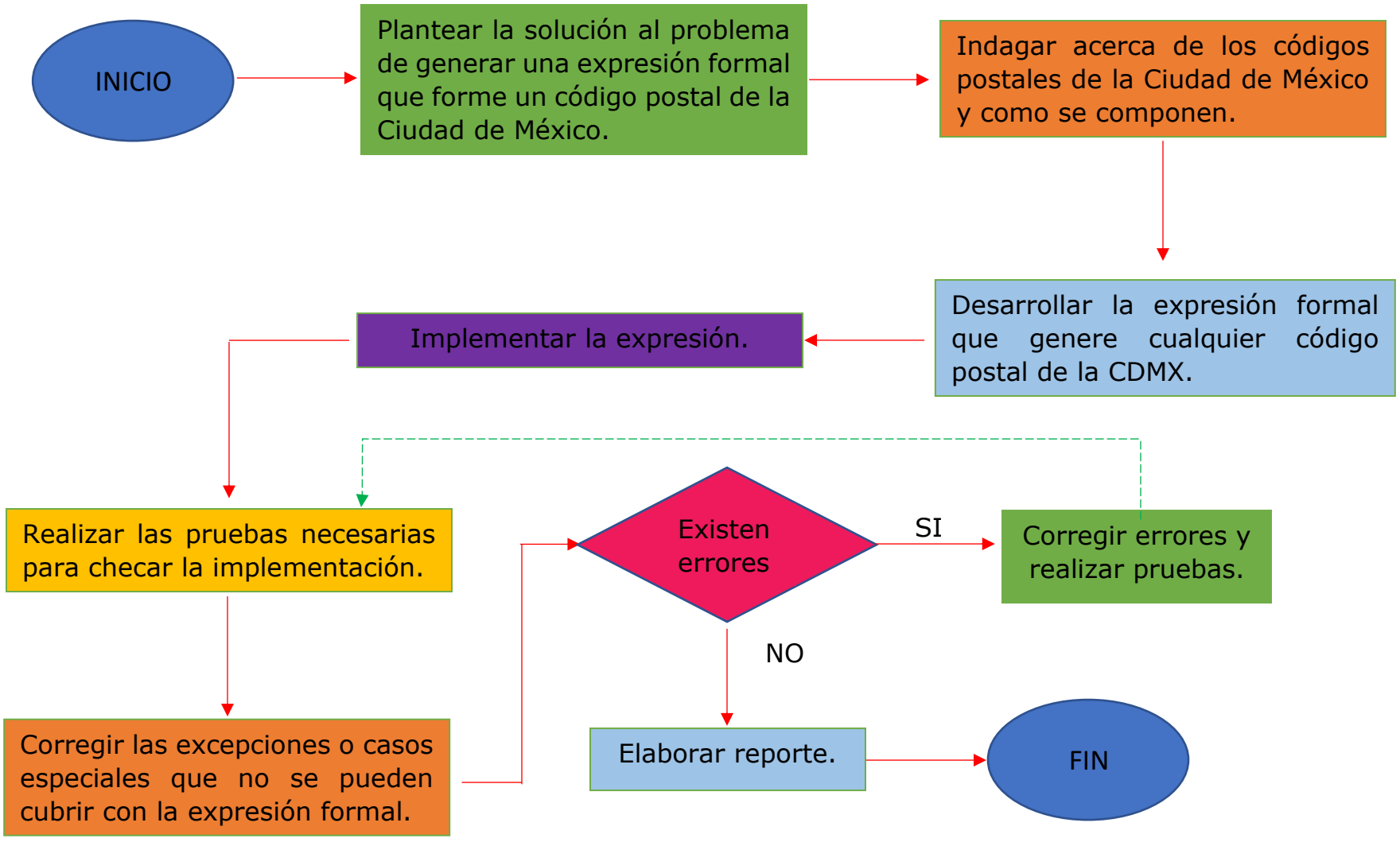
Realizar un programa que reconozca un lenguaje regular determinado a partir de una cadena ingresada por el usuario, depende demasiado de la expresión regular que se forme; ya que la implementación en cualquier lenguaje programación (de preferencia lenguaje C) es sencilla si la expresión regular está bien definida.

PLANTEAMIENTO DEL PROBLEMA

Generar la expresión regular para formar códigos postales de la Ciudad de México; una vez generada, realizar un programa que reconozca el lenguaje regular determinado. El programa deberá contar con las siguientes características:

- El programa tendrá como entrada una expresión regular determinada en un alfabeto, por ejemplo: Σ= {0,1} Σ= {a,b}, Σ={a-z}.
- La expresión regular puede ser cualesquiera, por ejemplo: 0(11)*0, ab(a|b)*.
- Con base en la expresión regular, el programa validará las cadenas que pertenezcan a la expresión regular, mismas que introducirá el usuario durante la ejecución del programa.

DIAGRAMA


INICIO
Plantear la solución al problema de generar una expresión formal que forme un código postal de la Ciudad de México.
Indagar acerca de los códigos postales de la Ciudad de México y como se componen.
Desarrollar la expresión formal que genere cualquier código postal de la CDMX.
Implementar la expresión.
Realizar las pruebas necesarias para checar la implementación.
Existen errores
SI
Corregir errores y realizar pruebas.
NO
Corregir las excepciones o casos especiales que no se pueden cubrir con la expresión formal.
Elaborar reporte.
FIN

IMPLEMENTACIÓN DE LA SOLUCIÓN

Este programa es relativamente sencillo, como primer paso programamos una función que valide en términos generales la cadena; es decir, las características principales de esta expresión dependen de la longitud de la cadena y el contenido de la misma, no pude tener una longitud mayor o menor de 5, debe ser estrictamente igual a 5 y no puede contener otro tipo de caracteres que no sean números enteros.

```
int verificar(char * cadena){
     int numero = 0, i = 0;
     if(strlen(cadena) == 5){ //Verificamos longitud de la cadena
          while(cadena[i] != '\0'){
               if(cadena[i]>='0' && cadena[i]<='9') i++;
               //Verificamos el contenido de la cadena
               else return 0;
          }
     return 1;
     }
     else return 0;
}
```

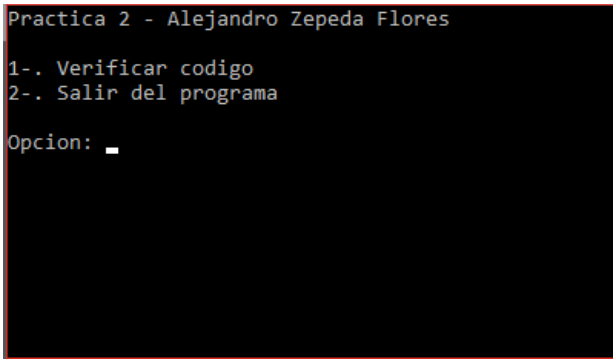
El siguiente paso corresponde a la validación de la expresión, una vez que se han cumplido las características previamente mencionadas. Como se pretende generar códigos postales de la Ciudad de México, los dos primeros caracteres deben estar contenidos dentro de un rango de ‘0-1’. Cumpliendo este requisito, se tendrá que verificar las posibles combinaciones que se puedan generar, por lo tanto, el siguiente caracteres dependerá del primero, si el primero es ‘0’ el segundo puede tomar un rango de ‘0-1’, si el primero es ‘1’ el segundo puede tomar un rango de ‘0-6’.

```
int validar(char * cadena){
     int check = 0, i = 0;
     if(cadena[0] == '0' || cadena[0] == '1')//Validación 1° digito
          if(cadena[1]>'0' && cadena[1]<='9') //Validación 2° digito
               if(cadena[0] == '1' && cadena[1]>'6') //Combinaión
                    return check;
               else check = 1;
          else return check;
     else return check;

}
```

Ahora trabajaremos un poco con a interfaz del programa, cumpliendo con los requerimientos del mismo, pero la a vez tratando de hacer más agradable para el usuario la interacción con el programa.
Como primer paso solicitamos al usuario el código postal que desea validar, después empezamos las validaciones con las funciones previamente programadas y le mostramos el mensaje de aprobación o negación, según sea el caso.
El siguiente punto es cumplir con el requerimiento de ingresar y verificar cadenas hasta que el usuario lo desee, esto se logra fácilmente con un ciclo while y es así como terminamos con la implementación del programa.

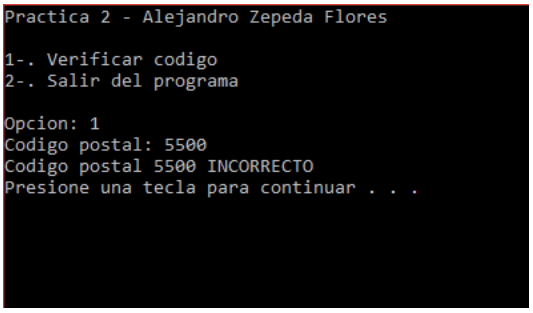
```
int main(int argc, char *argv[]) {
     int repeat = 0, opt = 0, i = 0;
     int * numero = (int *)malloc(sizeof(int)*5);
     char * codigo = (char *)malloc(sizeof(char)*5);

     do{
          printf("Practica 2 - Alejandro Zepeda Flores\n\n");
          printf("1-. Verificar codigo\n2-. Salir del programa\n\n");
          printf("Opcion: "); scanf("%d", &opt);
          switch(opt){
               case 1:
                    printf("Codigo postal: ");
                    fflush(stdin); gets(codigo);
                    if(verificar(codigo))
                    if(validar(codigo))
                    printf("Codigo postal %s CORRECTO\n",codigo);
                    else printf("Codigo postal %s ERRONEO\n",codigo);
                    elseprintf("Codigo postal %s ERRONEO\n",codigo);
                    system("pause"); system("cls"); repeat = 1;
               break;
               case 2:
                    repeat = 0;
               break;
               default:
                    repeat = 1;
                    system("cls");
          }
     }while(repeat == 1);
     return 0;
}
```

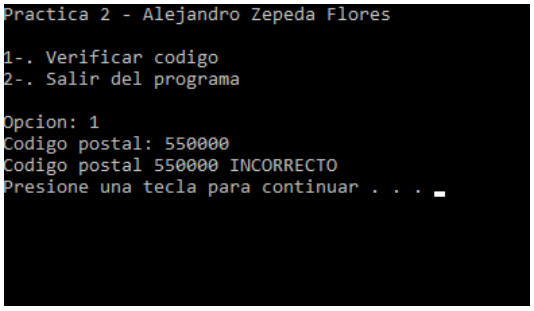
FUNCIONAMIENTO



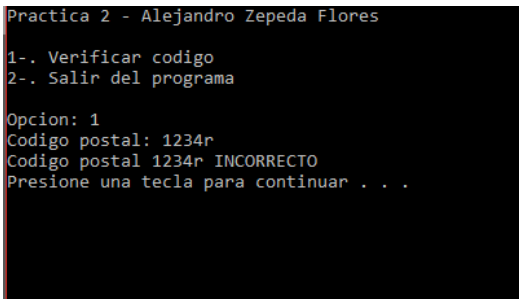
Interfaz de usuario

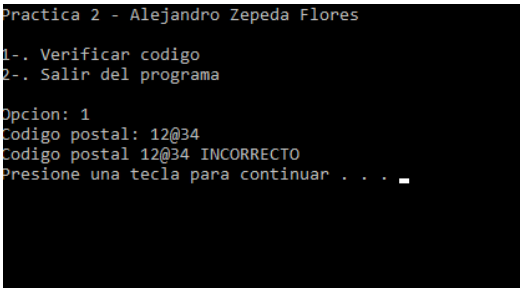
Primera operación – Verificar la longitud del código.



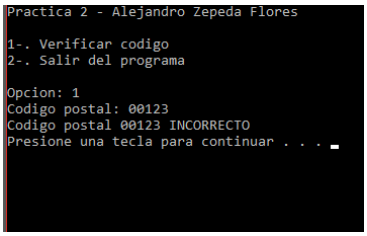


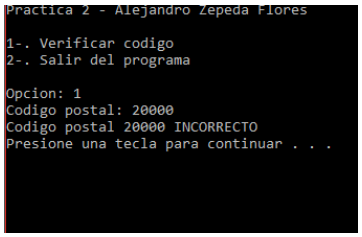
Segunda operación – Validación del contenido

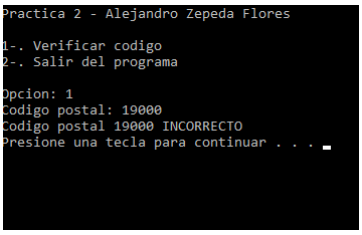




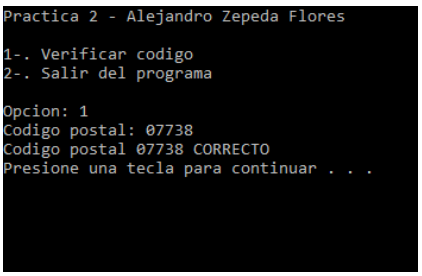
Tercera operación – Validación del código

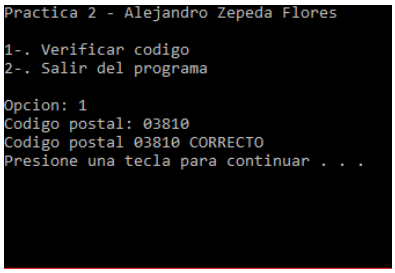


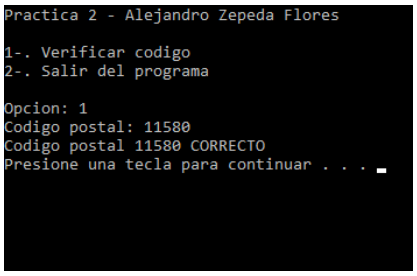




Códigos postales validos







CONCLUSIÓN

Esta práctica fue interesante porque me aclaro el punto de vista para generar expresiones formales y validarlas, relativamente es sencilla; implementarla si lo fue, es fácil desarrollar cualquier idea, lo complicado es generar esas ideas.

Formar un código postal fue un gran reto para desarrollar, en especial porque en un principio pensé que con puras validaciones en lenguaje podría resolverlo, pero dentro de la sección de prueba y error, noté que se generaban ciertas excepciones que no podía controlar con la expresión formal.

Como punto adicional a la conclusión de la práctica y como estudiante de ESCOM, para mí es mucho más complicado generar un reporte que cualquier expresión general.

BIBLIOGRAFÍA

- Mi Código Postal. (2018). Delegaciones de la Ciudad de México. marzo 18, 2018, de Correos de México Sitio web: http://micodigopostal.org/ciudad-de-mexico/
- 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
- Dirección Corporativa de Planeación Estratégica Gerencia de Sistemas de Información Geográfica . (2012). Dirección Corporativa de Planeación Estratégica Gerencia de Sistemas de Información Geográfica Noviembre2012 Manual de Asignación de Códigos Postales

y Estandarización de Domicilios Postales . Marzo 18,2018, de Correos de México Sitio web: http://correosdemexico.gob.mx/AcercaCorreos/NormatecaInterna/Documents/NormasInternas/mp_codigopostal_domicilios_a1.pdf