INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ALGORITMOS Y ESTRUCTURA DE DATOS

GRUPO: 2CV11

ALUMNO:

JONATHAN SAID GÓMEZ MARBÁN

PRÁCTICA 2:

“CURP”

Introducción

Para esta práctica requerimos conocimientos en apuntadores y estructuras, así que recordaremos algunos conceptos de ellos. Para la parte de estructura utilizaremos la palabra reservada typedef proporciona un mecanismo para la creación de sinónimos (o alias) para tipos de datos primitivos (int, char, float, etc.). Por ejemplo:

```
typedef struct {
        char c;
        int i;} Ejemplo;
```

Después de haber hecho una estructura como la que tenemos a lado, podremos crear variables de tipo Ejemplo. Tal y como se muestra en la siguiente imagen. `Ejemplo a[10];`

Tambien recordemos que si quisieramos acceder al contenido de un campo de a de tipo Ejemplo, seria de la siguiente forma: a -> c ó a -> i. Para crear un apuntador de a hacia alguno de campos sería, a.c ó a.i

Creo tambien oportuno mencionar en este apartado a la función strcpy, ya que será utilizada en repetidas ocasiones dentro del programa, con ella podremos copiar de una cadena de caracteres en otro cadena de caracteres; todos los caracteres que estaban en la cadena destino (la cadena en donde se esta poniendo la copia) desaparecen, aunque la cadena destino fuera más larga que la cadena fuente (la cadena que se copia).

Otra funnción que se usa dentro del programa es strncmp, la cual recibe dos cadenas, a y b, y un entero (el número máximo de caracteres que se van a comparar) y devuelve un entero. El entero que resulta de efectuar la llamada strncmp(a, b,n) codifica el resultado de la comparación:

- si el valor de retorno es <0, entonces indica que a es menor que b.
- si Valor de retorno> 0, indica que b es menor que a.
- si Valor de retorno = 0, indica que a es igual a b.

La función strlen esta función devuelve el total (entero) de caracteres que conforman una cadena.

Todas las funciones mencionadas con anterioridad nos las proporcionará librera estándar (string.h) y nos sirven para facilitar el manejo de cadenas de caracteres, así que haremos uso de ellas y sumado a lo explicado sobre el tema de estructuras, constituye la mayoría de las funciones que proporcionan algunos archivos de cabecera.

La función toupper, la cual devuelve la mayúscula asociada a caracter, si la tiene; si no, devuelve el mismo caracter. También ocupamos otra función que es tolower que funciona de manera similar a toupper, solo que, en vez de devolver una mayúscula asociada a carácter, devuelve un carácter.

Problemas relacionados al tema a los cuales te enfrentaste al programar la práctica.

Uno de los problemas principales que tuve fue, en el primer problema propuesto (introducir nombre), era identificar cuales eran las 3 palabras que me interesaban, es decir, el nombre1, el apellido1 y apellido2. Porque como sabemos el usuario puede meter hasta 255 caracteres, es, decir, podía con facilidad meter mas de 3 palabras (con palabras me refiero a que se que termina una y empieza otra con los espacios " "). En un principio pensaba solo tener un campo para el nombre y finalmente para resolver el ejercicio tuve que crear uno para el nombre, otro para el primer apellido y otro para el segundo apellido. De hecho, gracias a la lógica y una función que utilicé en el segundo problema (fecha de nacimiento), fue como también logré resolver el primer problema. Esta función a la que me refiero es strtok, la cual recibe una cadena y una constante de apuntador a carácter y básicamente se encarga de dividir la cadena en una serie de tokens usando un delimitador.
Cuando hablamos de tokens nos referimos a una parte de la cadena (una división de la cadena) y dicho delimitador es un carácter que nosotros sabemos que aparece en la cadena, por ejemplo, para el caso de una fecha con el formato DD-MM-YYYY, nuestro delimitador seria el carácter '-' y los tokens serían, DD, MM y YYYY.
Otro problema que tuve fue el manejo de los campos, el acceder a su valor, en un principio, al invocar las funciones en el main, solo apuntaba a el campo y así lograr que se guardaran los datos requeridos, o sea que en las funciones llegaba a tener 3 o más parámetros, después los cambie a que solo recibieran una variable de mi estructura, entonces al momento de invocar en la main, solo tendría que pasar una variable de tipo CURP. De nuevo (ya que en la practica 1 también ocurrió), tuve problemas para determinar el tamaño de algunos vectores al momento de inicializarlos y luego utilizar la función fgets para leerlos.

En general, esos fueron los problemas principales que tuve al realizar la práctica.

Documentación (capturas de pantalla).

```
#include <stdio.h> /*fgets*/
#include <string.h> /*strlen, strncmp, strcpy, strtok*/
#include <ctype.h>/*toupper, tolower*/
```

Inclusión de archivos cabecera. Imagen 1.

Incluimos los archivos cabeceras que utilizaremos, en los comentarios del programa se logra observar que funciones se ocupan de dichos archivos.

Se crea una estructura con typedef y la nombramos como CURP

```c
typedef struct
{
    char nombreCompleto[255];
    char nombre1[255];
    char apellido1[255];
    char apellido2[255];

    char fechaNacimiento[12];
    char fechaDia[3];
    char fechaMes[3];
    char fechaYear[3];
    char sexo;

    char estado[50];
    char estadoConvertido[3];
} CURP;
```

Dentro de la estructura tenemos 10 apuntadores a caracteres con memoria estática (vectores) y uno de tipo char. Todas ellas, nos ayudaran a trabajar con las funciones que resuelven los problemas propuestos y por supuesto para crear el CURP.

Estructura CURP. Imagen 2.

```c
/*introducirNombre, no regresa nada, pero recibe una variable de tipo apuntador a CURP
Se introduce y lee el nombre, se separa y se determina cual es el nombre, primer y segundo apellido
los valores de dichas variables se guardan en los campos correspondientes*/
void introducirNombre(CURP *curp)
{
    printf("\n1. Nombre (nombre1 nombre2 ... apellido1 apellido2): ");
    fgets(curp->nombreCompleto, 256, stdin);

    const char separacion[2] = " ";

    char *tokens[255];
    char *token;
    int numeroTokens = 0;
    token = strtok(curp->nombreCompleto, separacion);
    while (token != NULL)
    {
        tokens[numeroTokens] = token;
        token = strtok(NULL, separacion);
        numeroTokens++;
    }
    strcpy(curp->nombre1, tokens[0]);
    strcpy(curp->apellido1, tokens[numeroTokens - 2]);
    strcpy(curp->apellido2, tokens[numeroTokens - 1]);
    return;
}
```

Función introducirNombre. Imagen 3.

introducirNombre, no regresa nada, pero recibe una variable de tipo apuntador a CURP. En las primeras 2 líneas solicitamos que se ingrese el nombre, lo lee y lo guarda en el campo nombreCompleto. Las siguientes 10 líneas de código me permiten separar el nombre en

donde encuentre el valor " ", es donde será dividido y contamos cuantos tokens o divisiones se han realizado, para ello primero, declaramos el delimitador que nos ayudara a saber dónde dividir la cadena, que en este caso es el espacio o " ", posteriormente usamos una matriz (que guardara las cadenas separadas, los nombres y apellidos), un vector (que guarda una cadena, como nombre o apellido) y con un entero contaremos cuantas veces fue separada la cadena, mientras que se van guardando las cadenas de nombres y apellidos. Que token sea diferente de NULL significa que la cadena todavía no llega a su fin. Luego se toma el primer nombre (posición 0), primer apellido(número de divisiones menos 2) y segundo apellido(número de divisiones menos 1) y se lo asignamos a los campos nombre1, apellido1 y apellido 2 respectivamente.

```c
/*intoducirFecha, no regresa nada y recibe un parametro de tipo apuntador a CURP
Se introduce y lee la fecha, se secciona por dia mes y año, posteriormente se gaurdan en
los campos correspondientes, a la sepraracion del año todavía se le hacen modificaciones para
que me de los 2 digitos que interesan y se guarda en el campo correspondiente*/
void introducirFecha(CURP *curp)
{
    printf("\n2. Fecha de nacimiento (introduce con el siguiente formato: DD-MM-YYYY: ");
    const char separacion[2] = "-";
    char fecha[12];
    char year[6];
    fgets(fecha, 12, stdin);
    strcpy(curp->fechaDia, strtok(fecha, separacion));
    strcpy(curp->fechaMes, strtok(NULL, separacion));
    strcpy(year, strtok(NULL, separacion));
    curp->fechaYear[0] = year[2];
    curp->fechaYear[1] = year[3];
    curp->fechaYear[2] = '\0';
}
```

Función introducirFecha. Imagen 4.

intoducirFecha, no regresa nada y recibe un parámetro de tipo apuntador a CURP. En las primeras 5 líneas de código, se pide que ingrese la fecha con un formato especifico, luego se crea el delimitador que usaremos para la función strtok y el vector donde guardamos la fecha, también se crea un vector para guardar el año, una vez ya este separada la cadena, los valores ya separados del día y mes se guardan en el campo de fechaDia y fechaMes respectivamente del parámetro de la función.

```c
/*introducirSexo, no regresamos nada y recibimos como parametro un apuntador a CURP
se introduce, lee y guarda el sexo en el campo correspondiente.*/
void introducirSexo(CURP *curp)
{
    printf("\n3. Sexo: (H Hombre, M mujer): ");
    fgets(&(curp->sexo), 3, stdin);
}
```

Función introducirSexo. Imagen 5.

introducirSexo, no regresamos nada y recibimos como parámetro un apuntador a CURP. Se imprime en pantalla el formato con el que se requiere se meta el sexo, después se lee y para ello, hacemos un apuntador a el acceso al campo de sexo del parámetro y dicho valor introducido se guarda en este campo.

```
/*convertirCadenaMinuscula, recibe un apuntador a caracter y regresa un apuntador a caracter
Se recibe una cadena y la funcion la regresa toda en minusculas.*/
char *convertirCadenaMinuscula(char *cadena)
{
    for (int i = 0; cadena[i] != '\0'; i++)
    {
        cadena[i] = tolower(cadena[i]);
    }
    return cadena;
}
```

Función convertirCadenaMinuscula. Imagen 6.

convertirCadenaMinuscula, recibe un apuntador a carácter y regresa un apuntador a carácter. Para convertir toda una cadena introducida, utilizaremos otra función ya establecida en un archivo cabecera llamada tolower y con un for recorreremos toda la cadena dada hasta que llegue a el fin de cadena que identificamos con "\0", iteramos en i y vamos asignando la minúscula de esa posición en la misma cadena, al final regresamos la cadena.

```c
/*introducirEntidadFederativa, regresa un entero, y recibe como parametros un apuntador a CURP
Se introduce la entidad federativa, si es correcta me regresa la breviacion correspondiente, sino es correcta
regresa un -1*/
int introducirEntidadFederativa(CURP *curp)
{
    printf("\n4. Introduce entidad federativa: ");
    fgets(curp->estado, 52, stdin);

    char *entidadFederativaMinusculas;
    entidadFederativaMinusculas = convertirCadenaMinuscula(curp->estado);
    int numeroCaracteres = strlen(entidadFederativaMinusculas);

    if (strncmp(entidadFederativaMinusculas, "aguascalientes\n", numeroCaracteres) == 0)
    {
        strcpy(curp->estadoConvertido, "AS\0");
        return 0;
    }
    else if (strncmp(entidadFederativaMinusculas, "baja california\n", numeroCaracteres) == 0)
    {
        strcpy(curp->estadoConvertido, "BC\0");
        return 0;
    }
    else if (strncmp(entidadFederativaMinusculas, "baja california sur\n", numeroCaracteres) == 0)
    {
        strcpy(curp->estadoConvertido, "BS\0");
        return 0;
    }
    else if (strncmp(entidadFederativaMinusculas, "campeche\n", numeroCaracteres) == 0)
    {
        strcpy(curp->estadoConvertido, "CC\0");
        return 0;
    }
    else if (strncmp(entidadFederativaMinusculas, "coahuila\n", numeroCaracteres) == 0)
    {
```

Función introducirEntidadFederativa. Imagen 7.

Cabe mencionar que no se sacara captura a toda la función, ya que es innecesario y es demasiado larga y algo repetitiva. La función introducirEntidadFederativa, regresa un entero, y recibe como parámetros un apuntador a CURP. Leemos la entidad federativa y la guardamos en el campo estado del parámetro, ya que accedemos a su valor, luego se crea un apuntador a carácter y usaremos la función convertirCadenaMinuscula que ya hemos definido con anterioridad y le pasaremos como parámetro la cadena leída y el resultado (la cadena pero toda en minúsculas) lo guardamos en el apuntador creado, esto lo hacemos con el fin de que cuando el usuario introduzca la entidad federativa, sea indiferente si introduce minúsculas o mayúsculas, porque al momento de hacer la comparación de cadenas con una cadena (entidad federativa en minúsculas), nos dé como resultado si son iguales las cadenas o no. Entonces una vez teniendo la cadena en minúsculas contamos cuantos caracteres tiene con strlen y lo almacenamos en numeroCaracteres que es de tipo entero. A través de la función strncpm introducimos la cadena que esta toda en minúsculas, luego entre comillas con alguna de las 33 entidades federativas propuestas para el programa y se nota que a todas se le agrega un salto de línea, esto es porque cuando se lee

la cadena que introduce el usuario, al momento de presionar enter, fgets también lo lee y es por eso que se compara de esa forma, luego, el número máximo de caracteres que se van a comparar es igual a la variable numeroCaracteres, si la función strncmp resulta ser igual a 0 significa que las cadenas son igual carácter a carácter, por lo que con strcpy, accedemos al valor del campo estadoConvertido del parámetro y le asignamos la abreviación que le corresponde a ese estado y salimos de la función. (nótese que todas las abreviaciones tienen \0 que me representa el fin de cadena, esto lo hacemos cona la finalidad de que al imprimir se reconozca que ya se terminó la cadena y no imprima basura). Si resulta que no es igual la cadena que se compara con ninguna de las entidades federativas, entonces imprimimos "intenta de nuevo" y regresamos un -1, que después se trabajará con ello en la main.

```
/*extraerConsonante, regresa un caracter y recibe un apuntador a caracter y un entero
Me permite saber cual es la n consonante en una cadena y me regresa la consonante, si no existe
la consonante deseada regresa un espacio.*/
char extraerConsonante(char *cadena, int posicion)
{
    int numeroConsonantes = 1;
    for (int i = 0; cadena[i] != '\0'; i++)
    {
        char actual = tolower(cadena[i]);
        if (actual != 'a' && actual != 'e' && actual != 'i' && actual != 'o' && actual != 'u')
        {
            if (numeroConsonantes == posicion)
            {
                return actual;
            }
            numeroConsonantes++;
        }
    }
    return ' ';
}
```

Función extraer consonante. Imagen 8.

extraerConsonante, regresa un carácter y recibe un apuntador a carácter y un entero. Declaramos un entero y lo inicializamos en 1, este nos servirá para saber cuántas consonantes se han encontrado. Luego, con un for que va desde 0 hasta el fin de la cadena (parámetro), comparamos si el carácter en la posición i es diferente a una vocal, para ello, transformamos el carácter en minúscula con tolower, para así no tener problemas en saber si se ha introducido una mayúscula, si resulta ser diferente de las vocales, entonces me regresa ese carácter, sino se suma 1 al número de consonantes. El parámetro posición nos servirá para determinar si ya ha habido una consonante antes (la primera letra o segunda del apellido1, la primera letra del apellido2 o la primera letra del nombre).

```c
/*crearCurp, regresa nada y recibe un apuntador a CURP
Imprime todos los elmentos que se requieren para el CURP*/
void crearCurp(CURP *curp)
{
    printf("%c", toupper(curp->apellido1[0]));
    printf("%c", toupper(curp->apellido1[1]));
    printf("%c", toupper(curp->apellido2[0]));
    printf("%c", toupper(curp->nombre1[0]));
    printf("%s", curp->fechaYear);
    printf("%s", curp->fechaMes);
    printf("%s", curp->fechaDia);
    printf("%c", toupper(curp->sexo));
    printf("%s", curp->estadoConvertido);

    char primerLetra = tolower(curp->apellido1[0]);
    if (primerLetra == 'a' || primerLetra == 'e' || primerLetra == 'i' || primerLetra == 'o' || primerLetra == 'u')
    {
        printf("%c", toupper(extraerConsonante(curp->apellido1, 1)));
    }
    else
    {
        printf("%c", toupper(extraerConsonante(curp->apellido1, 2)));
    }

    primerLetra = tolower(curp->apellido2[0]);
    if (primerLetra == 'a' || primerLetra == 'e' || primerLetra == 'i' || primerLetra == 'o' || primerLetra == 'u')
    {
        printf("%c", toupper(extraerConsonante(curp->apellido2, 1)));
    }
    else
    {
        printf("%c", toupper(extraerConsonante(curp->apellido2, 2)));
    }

    primerLetra = tolower(curp->nombre1[0]);
    if (primerLetra == 'a' || primerLetra == 'e' || primerLetra == 'i' || primerLetra == 'o' || primerLetra == 'u')
    {
        printf("%c", toupper(extraerConsonante(curp->nombre1, 1)));
    }
    else
    {
        printf("%c", toupper(extraerConsonante(curp->nombre1, 2)));
    }

    printf("%s", "00");
}
```

Función crearCurp. Imagen 9.

crearCurp, regresa nada y recibe un apuntador a CURP. Como se logra notar aquí es donde imprimimos uno por uno, los valores que hemos ido sacando en cada variable para imprimir el CURP, las variables que requieran estar en mayúscula como la primera letra de cada apellido, nombre, etc. las haremos mayúsculas con la función toupper, como observamos tenemos if's para los casos de la consonante del primer apellido1, apellido2 y nombre, esto es porque habrá casos donde la primera letra de cualquiera de estos sea una consonante. Entonces comparamos en las primeras posiciones de cada campo.

```c
int main()
{
    CURP usuario;
    printf("\n\t\t:::Datos del usuario:::");
    introducirNombre(&usuario);
    introducirFecha(&usuario);
    introducirSexo(&usuario);
    /*Aquí se verifica que la entidad federativa introducida
    haya sido la correcta*/
    int error = 0;
    do {
        error = introducirEntidadFederativa(&usuario);
    } while (error != 0);
    /*Aquí se imprime el CURP*/
    crearCurp(&usuario);

    return 0;
}
```

Función main. Imagen 11.

La función main donde mandamos a llamar todas las funciones que requerimos para imprimir el CURP.

Procederemos a presentar las capturas de las pruebas propuestas en la práctica.

```
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>gcc CURP.c

C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>a

                :::Datos del usuario:::
1. Nombre (nombre1 nombre2 ... apellido1 apellido2): Ricardo Angel Esteban Almazan Perez

2. Fecha de nacimiento (introduce con el siguiente formato: DD-MM-YYYY: 21-03-1970

3. Sexo: (H Hombre, M mujer): H

4. Introduce entidad federativa: DisTriTO FEDERaL
ALPR700321HDFLRC00
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>
```

Prueba 1. Imagen 12.

Como se logra observar introdujimos todos los datos que me fueron proporcionados en la practica para la prueba 1 y ese es el resultado. En efecto, es el CURP de dicha persona.

```
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>gcc CURP.c

C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>a

               :::Datos del usuario:::
1. Nombre (nombre1 nombre2 ... apellido1 apellido2): Jonathan Said Gomez Marban

2. Fecha de nacimiento (introduce con el siguiente formato: DD-MM-YYYY: 23-10-2001

3. Sexo: (H Hombre, M mujer): H

4. Introduce entidad federativa: mexico
GOMJ011023HMCMRN00
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>
```

Mi CURP. Imagen 13.

Como se logra observar introduje todos mis datos y, en efecto, es mi CURP.

CURP real. Imagen 14.

```
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>gcc CURP.c

C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>a

               :::Datos del usuario:::
1. Nombre (nombre1 nombre2 ... apellido1 apellido2): Teresa Angelica Berenice Briseño Concepcion

2. Fecha de nacimiento (introduce con el siguiente formato: DD-MM-YYYY: 30-11-1981

3. Sexo: (H Hombre, M mujer): M

4. Introduce entidad federativa: Baja CALIFORNIA sur
BRCT811130MBSRNR00
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>
```

CURP Teresa. Imagen 15.

Por último, introducimos los datos de Teresa y verificamos que todo esta correcto.

Conclusiones

Con esta práctica aprendí a tener un mejor manejo de cadenas, a partir de funciones como strncmp, strcpy, entre otras; antes de esta práctica no sabía que existían, pero al momento de investigar y ver como funcionaban entendí mejor como usarlas y en que casos. También me di cuenta de que mi manejo con estructuras mejoro ya que ahora que he practicado mas con ellas, en como crearlas, acceder al valor de cierto campo y ponerlas como parámetro, me ayudo a entender mejor. Pude entender una función que en un principio me parecía muy complicada la cual es strtok, cuando estaba buscando como resolver el problema y la encontré, sin embargo, en un prncipio la descarte por parecer muy complicada, pero me di cuenta de lo que hacia y pude implementarla dentro de mi programa.

Bibliografías

[1] (s.f.) "El lenguaje C" Disponible en https://www.fing.edu.uy/tecnoinf/mvd/cursos/prinprog/material/teo/prinprog-teorico08.pdf

[2] (s.f.) "Función de biblioteca C - strtok ()" Disponible en https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm

[3] (s.f.) "Función de biblioteca C - strncmp () Disponible en https://www.tutorialspoint.com/c_standard_library/c_function_strncmp.htm