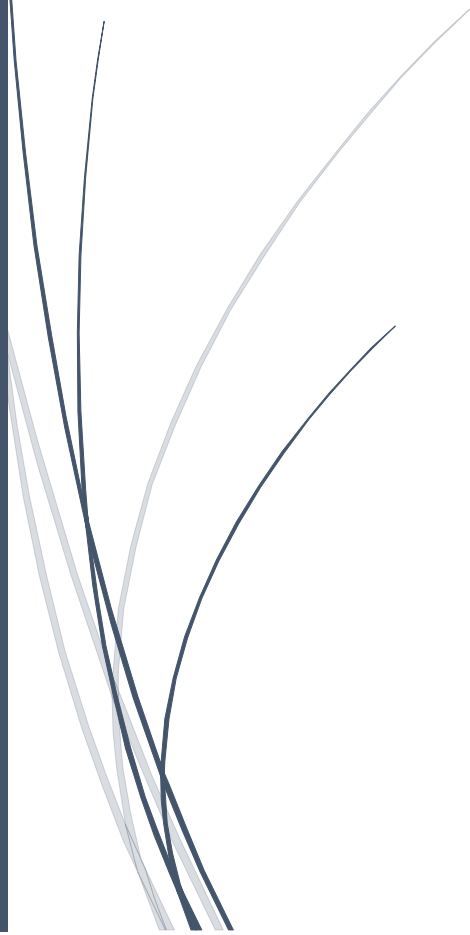


2/6/2022

# Hashing

Practica Tema 4



Alejandro Colmenero Moreno  
ALGORITMIA

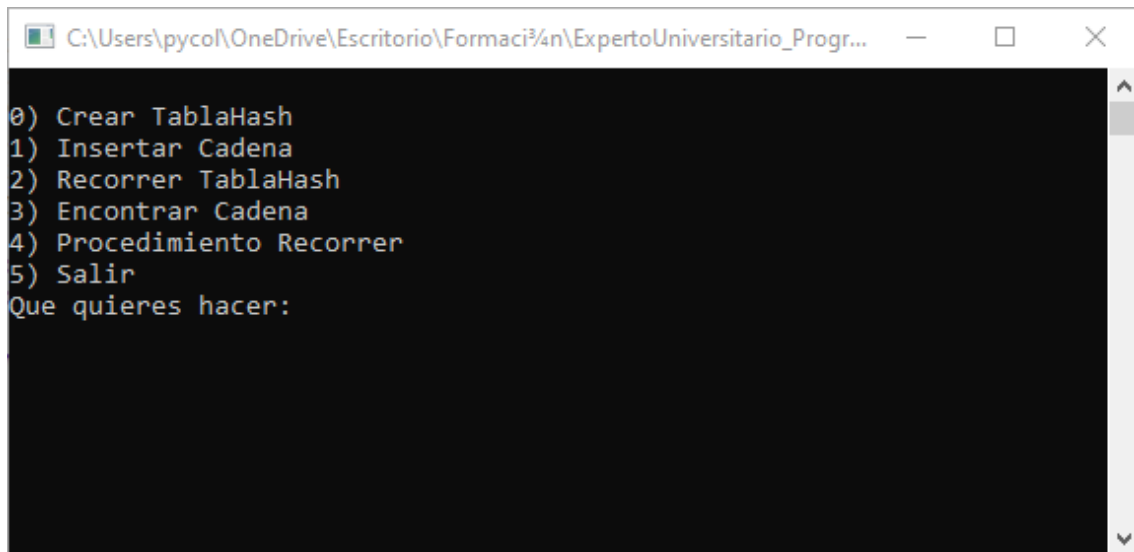
## ÍNDICE

EJERCICIO .....	2
EXPLICACIÓN .....	2
ERRORES.....	4
CÓDIGO .....	5
BIBLIOGRAFÍA.....	10

## EJERCICIO

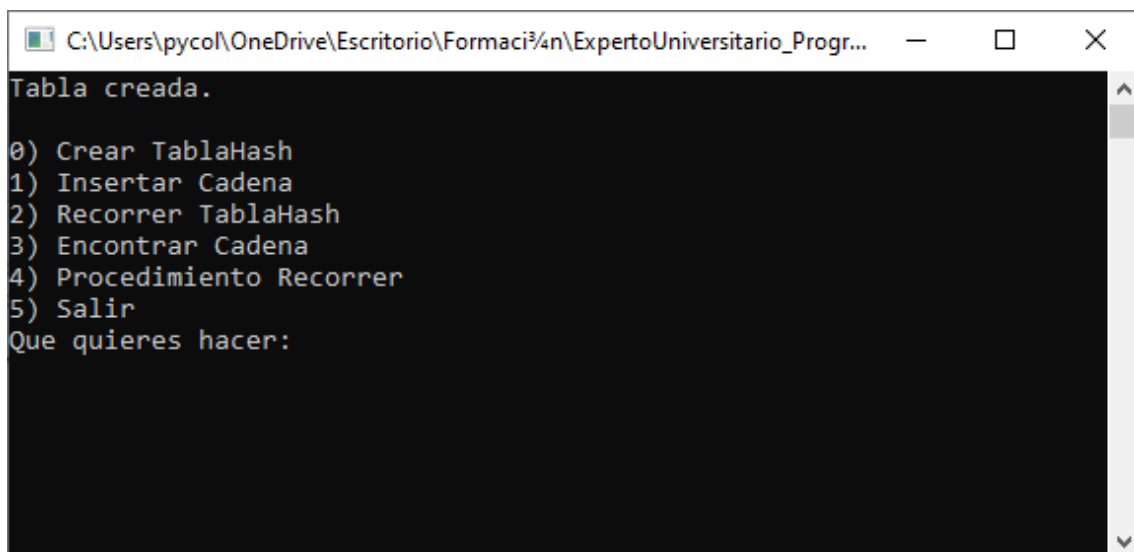
### EXPLICACIÓN

El proyecto se compone de un menú que te pide una orden.



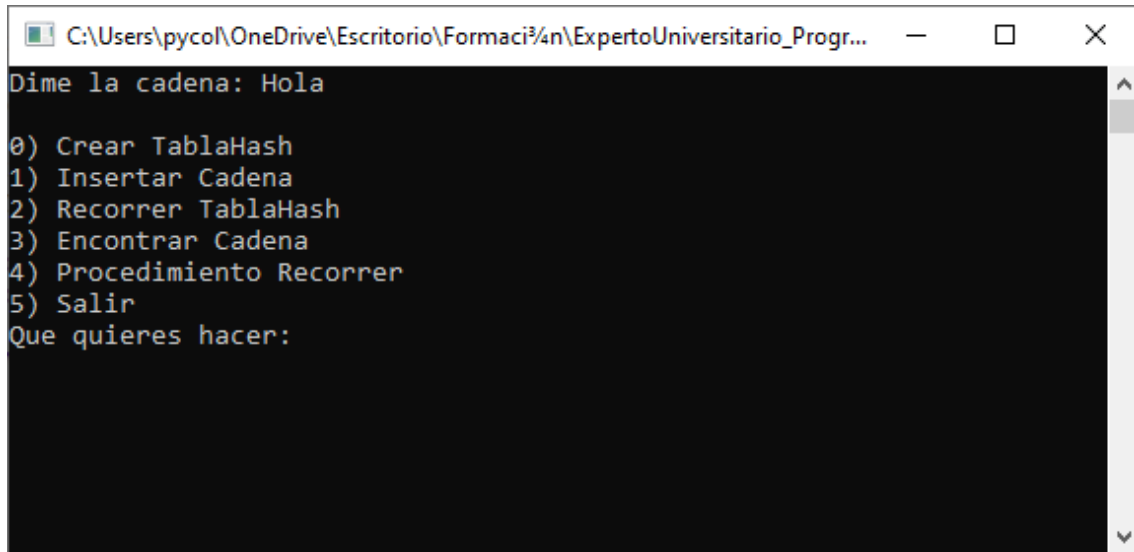
```
C:\Users\pyco\OneDrive\Escritorio\Formaci3/4n\ExpertoUniversitario_Progr... — □ ×  
0) Crear TablaHash  
1) Insertar Cadena  
2) Recorrer TablaHash  
3) Encontrar Cadena  
4) Procedimiento Recorrer  
5) Salir  
Que quieres hacer:
```

Si pulso 0, lo que hará es generar una nueva TablaHash y si todo ha ido bien te dice que se ha creado la tabla.



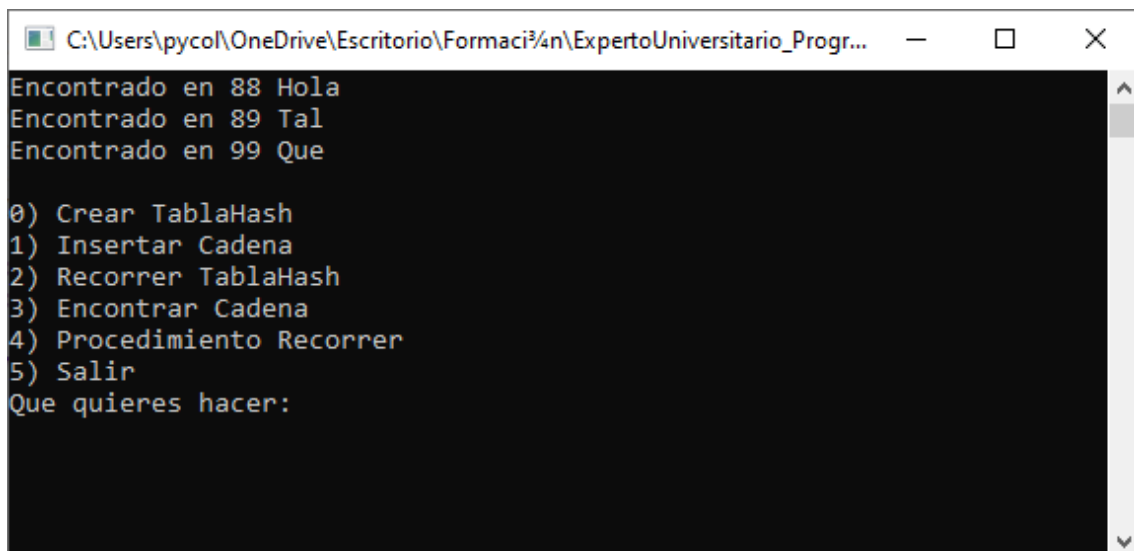
```
C:\Users\pyco\OneDrive\Escritorio\Formaci3/4n\ExpertoUniversitario_Progr... — □ ×  
Tabla creada.  
0) Crear TablaHash  
1) Insertar Cadena  
2) Recorrer TablaHash  
3) Encontrar Cadena  
4) Procedimiento Recorrer  
5) Salir  
Que quieres hacer:
```

Si pulso 1, me pide que escriba un texto y lo inserta en la tabla.



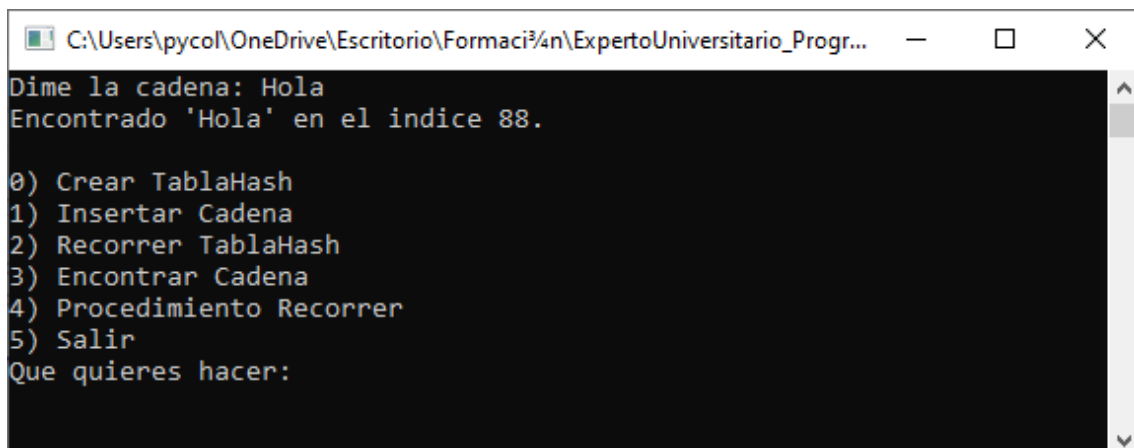
```
C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
Dime la cadena: Hola
0) Crear TablaHash
1) Insertar Cadena
2) Recorrer TablaHash
3) Encontrar Cadena
4) Procedimiento Recorrer
5) Salir
Que quieres hacer:
```

Pulsando 2, me muestra todos los registros de la tabla y sus índices:



```
C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
Encontrado en 88 Hola
Encontrado en 89 Tal
Encontrado en 99 Que
0) Crear TablaHash
1) Insertar Cadena
2) Recorrer TablaHash
3) Encontrar Cadena
4) Procedimiento Recorrer
5) Salir
Que quieres hacer:
```

En cambio, pulsando 3, me pedirá una palabra y la buscará en el hash. Si no existe lo indicará.



```
C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
Dime la cadena: Hola
Encontrado 'Hola' en el indice 88.
0) Crear TablaHash
1) Insertar Cadena
2) Recorrer TablaHash
3) Encontrar Cadena
4) Procedimiento Recorrer
5) Salir
Que quieres hacer:
```

```

C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
Dime la cadena: holaa
No se ha encontrado

0) Crear TablaHash
1) Insertar Cadena
2) Recorrer TablaHash
3) Encontrar Cadena
4) Procedimiento Recorrer
5) Salir
Que quieres hacer:

```

Por último, pulsando 4, hará el recorrido, pero si se topa con una celda vacía lo mostrará como “No asignado”:

```

C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
No asignado
No asignado
No asignado
No asignado
No asignado
Encontrado en 88 Hola
Encontrado en 89 Tal
No asignado
No asignado
No asignado
No asignado
No asignado

```

## ERRORES

Si intentas hacer cualquier cosa sin crear antes la tabla, te sale un error.

```

C:\Users\pycol\OneDrive\Escritorio\Formaci3\4n\ExpertoUniversitario_Progr...
Tabla no creada.

0) Crear TablaHash
1) Insertar Cadena
2) Recorrer TablaHash
3) Encontrar Cadena
4) Procedimiento Recorrer
5) Salir
Que quieres hacer:

```

## CÓDIGO

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NCASILLAS 100
#define VACIO NULL

static char* BORRADO = "";
typedef char **TablaHash;

// Declaración de funciones

void error(char*);
TablaHash crearTablaHash();
void borrarTablaHash(TablaHash);
int Hash(char*);
int Localizador(char*, TablaHash);
int Localizador1(char*, TablaHash);
int MiembroHash(char*, TablaHash);
void InsertarHash(char*, TablaHash);
void BorrarHash(char*, TablaHash);

int main()
{
    TablaHash t = NULL;
    /*t = crearTablaHash();

    printf("Hola %d\n", Hash("Hola"));
    printf("hola %d\n", Hash("hola"));
    InsertarHash("Hola",t);
    printf(":%s\n", t[Localizador("Hola",t)]);
    printf(":%s\n", t[Localizador("hola",t)]);*/

    char nombre[10];
    int menu = 0;
    while(menu != 5){

        printf("\n");
        printf("0) Crear TablaHash\n");
        printf("1) Insertar Cadena\n");
        printf("2) Recorrer TablaHash\n");
        printf("3) Encontrar Cadena\n");
        printf("4) Procedimiento Recorrer\n");

        printf("5) Salir\n");
    }
}

```

```

printf("Que quieres hacer: ");
scanf("%i", &menu);
fseek(stdin,0,SEEK_END);
printf("\n");
system("cls");

int aux = 0;
int ini = 0;

switch(menu){
    case 0:
        t = crearTablaHash();
        printf("Tabla creada.\n");
        break;
    case 1:
        // PEDIR DATOS
        if(t == VACIO) {
            error("Tabla no creada.\n");
        } else {
            printf("Dime la cadena: ");
            scanf("%s", &nombre);
            fseek(stdin,0,SEEK_END);
            InsertarHash(nombre,t);
        }

        break;
    case 2:

        if(t == VACIO) {
            error("Tabla no creada.\n");
        } else {

            for(int i=0;i<NCASILLAS;i++){
                aux=(ini+i)%NCASILLAS;
                if(t[aux] != NULL){
                    printf("Encontrado en %i %s\n",aux, t[aux]);
                }
            }

        }
        break;
    case 3:

        if(t == VACIO) {
            error("Tabla no creada.\n");
        } else {
            printf("Dime la cadena: ");
            scanf("%s", &nombre);

```

```

        if(t[Localizador(nombre,t)] != NULL){
            int localizador = Localizador(nombre,t);
            printf("Encontrado '%s' en el indice %i.\n",
t[localizador],localizador);
        } else {
            printf("No se ha encontrado\n");
        }
    }

    break;
case 4:
    if(t == VACIO) {
        error("Tabla no creada.\n");
    } else {
        ProdecimientoRecorrer(t);
    }
    break;
case 5:
    printf("Bye :)");
    break;
default:
    printf("\nNo te he entendido.\n\n");
}
}

return 0;
}

void error(char* msg){
    printf("%s\n",msg);
}

void ProdecimientoRecorrer(TablaHash t){
    int ini,aux;
    for(int i=0;i<NCASILLAS;i++){
        aux=(ini+i)%NCASILLAS;
        if(t[aux] != NULL){
            printf("Encontrado en %i %s\n",aux, t[aux]);
        } else {
            printf("No asignado\n");
        }
    }
}

TablaHash crearTablaHash(){
    TablaHash t;
    register int i;

    t = (TablaHash)malloc(NCASILLAS*sizeof(char*));

```



```

    if(t==NULL){
        error("Memoria insuficiente");
    }
    for(i = 0; i < NCASILLAS; i++){
        t[i]=VACIO;
    }
    return t;
}

void borrarTablaHash(TablaHash t){
    register int i;
    for(i = 0; i < NCASILLAS; i++){
        if(t[i]!=VACIO&& t[i]!=BORRADO){
            free(t[i]);
        }
    }
}

int Hash(char* cad){

    int valor;
    char *c;
    for(c=cad, valor=0; *c; c++){
        valor += (int)*c;
    }
    return valor%NCASILLAS;
}

int Localizador(char*x, TablaHash t){
    int ini,i,aux;
    ini=Hash(x);

    for(i=0; i<NCASILLAS; i++){
        aux=(ini+i)%NCASILLAS;
        //printf("\n%i",aux);
        if(t[aux]==VACIO){
            return aux;
        }
        if(!strcmp(t[aux],x)){
            return aux;
        }
    }

    return ini;
}

int Localizador1(char* x, TablaHash t){

    int ini,i,aux;
    ini=Hash(x);
    for(i=0; i<NCASILLAS; i++){
        aux=(ini+i)%NCASILLAS;

```

```

        if(t[aux]==VACIO || t[aux]==BORRADO){
            return aux;
        }
        if(!strcmp(t[aux],x)){
            return aux;
        }
    }
    return ini;
}

int MiembroHash(char* cad, TablaHash t){
    int pos = Localizador(cad,t);
    if(t[pos]==VACIO){
        return 0;
    } else {
        return !strcmp(t[pos],cad);
    }
}

void InsertarHash(char* cad, TablaHash t){
    int pos;
    if(!cad){
        error("Cadena inexistente");
    }
    if(!MiembroHash(cad,t)){
        pos=Localizador1(cad,t);
        if(t[pos]==VACIO || t[pos]==BORRADO){
            t[pos]=(char*)malloc((strlen(cad)+1)*sizeof(char));
            strcpy(t[pos],cad);
        } else {
            error("Tabla llena");
        }
    }
}

void BorrarHash(char* cad, TablaHash t){
    int pos = Localizador(cad,t);
    if(t[pos]!=VACIO && t[pos]!=BORRADO){
        if(!strcmp(t[pos],cad)){
            free(t[pos]);
            t[pos]=BORRADO;
        }
    }
}
}

```

## BIBLIOGRAFÍA

N/A