# PROGRAMACION ESTRUCTURADA
## Actividad 9

**Librerías en c, métodos de ordenación y búsqueda**

Héctor Daniel Camacho López

Matricula: 372239

Grupo 432

UABC (Universidad Autónoma de Baja California)

Facultad de ingeniería y diseño

Profesor: Pedro Nuñez Yepiz


Ensenada, Baja California a 09 de Octubre de 2023

CAPTURAS DE LIBRERÍA

```c
//Hector Daniel Camacho Lopez     372239                                          //
//Ensenada, Baja California a 20 de Septiembre del 2023                           //
//Libreria                                                                        //
//Nombre de la actividad: HDCL_ACT9_01_432                                        //

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define M 15
#define N 4

// PROTOTIPOS DE FUNCIONES //

void mayusculas(char cadena[60]);
void minusculas(char cadena[60]);
void capital(char cadena[60]);
int tamaño(char cadena[60]);
void voltear(char cadena[60]);
void noespacios(char cadena[60]);
char permitir(char cadena[60]);
void todas(char cadena[60]);
void palindromo(char cadena[]);

int valida_num(int ri, int rf);
bool no_repetir(int vect[], int n);
void llenar_vect_aleatorio(int vect[], int m, int ri, int rf);
void llenar_matriz(int matriz[4][4], int m, int ri, int rf);
void imprimir_mat(int matriz[4][4], int m, int n);
void imprimir_vect(int vect[], int m);
bool no_repetir_m(int matriz[4][4],int n);
void ordenar(int vect[], int m);
int busq_sec(int vect[], int n, int num);
void buscar (int vect[], int m);



// FUNCION DE MAYUSCULAS //
void mayusculas(char cadena[60])
{
    int i;
    printf("\n");
    for(i = 0; cadena[i] != '\0'; i++)
    {
        if (cadena[i] >= 97)
        {
            if (cadena[i] <= 122)
            {
                cadena[i] = cadena[i] - 32;
```

```
                {
                    cadena[i] = cadena[i] - 32;
                    printf("%c" ,cadena[i]);
                }
            }
        }
    }

    // FUNCION DE MINUSCULAS //
    void minusculas(char cadena[60])
    {
        int i;
        printf("\n");
        for(i = 0; cadena[i] != '\0'; i++)
        {
            if (cadena[i] >= 65)
            {
                if (cadena[i] <= 90)
                {
                    cadena[i] = cadena[i] + 32;
                    printf("%c" ,cadena[i]);
                }
            }
        }
    }

    // FUNCION QUE CONVIERTE CADENA EN "CAPITAL" //
    void capital(char cadena[60])
    {
        int i;
        printf("\n");
        cadena[0] = 67;
        cadena[1] = 65;
        cadena[2] = 80;
        cadena[3] = 73;
        cadena[4] = 84;
        cadena[5] = 65;
        cadena[6] = 76;
        cadena[7] = '\0';
        for(i = 0; cadena[i] != '\0'; i++)
        {
            printf("%c", cadena[i]);
        }
    }

    // FUNCION QUE CUENTA LOS CARACTERES DE LA CADENA //
    int tamaño(char cadena[60])
    {
        printf("\n");
        int i;
```

```
    printf("\n");
    int i;
    for(i = 0; cadena[i] != '\0'; i++);
    return i;
}

// FUNCION QUE VOLTEA LA CADENA AL REVES //
void voltear(char cadena[60])
{
    int i, t;
    printf("\n");
    t = tamaño(cadena);
    for (i = t-1; i >= 0; i--)
    {
        printf("%c", cadena[i]);
    }
}

// FUNCION QUE ELIMINA LOS ESPACIOS DE LA CADENA //
void noespacios(char cadena[60])
{
    int i, t, aux, espacio;
    printf("\n");
    t = tamaño(cadena);
    for(i = 0; i < t; i++)
    {
        if (cadena[i] == 32)
        {
            espacio = i;
            aux = i;
            while (cadena[aux] == 32 && aux < t - 1)
            aux ++;
            cadena[espacio] = cadena[aux];
            cadena[aux] = 32;
        }
    }

    for(i = 0; cadena[i] != '\0'; i++)
    {
        printf("%c", cadena[i]);
    }
}

// FUNCION QUE PERMITE O NO LA CADENA //
char permitir(char cadena[60])
{
    int i, t;
    t = tamaño(cadena);
    printf("\n");
```

```c
    t = tamaño(cadena);
    printf("\n");
    for(i = 0; cadena[i] != '\0'; i++)
    {

    if (cadena[0] == ' ' || cadena[t] == ' ')
    {
        printf("\nNO PERMITIDA, debe ser una cadenas sin espacios al inicio ni al final");
        cadena[i] = '\0';
    }

    if (cadena[i] == 32 && cadena[i-1] == 32)
    {
        printf("NO PERMITIDO, debe ser cadena sin espacios dobles");
        cadena[i] = '\0';
    }

    if (cadena[i] != 32 && (cadena[i] < 65 || cadena[i] > 122))
    {printf("\nNO PERMITIDO, solo caracteres alfabeticos");
        cadena[i] = '\0';
    }

    }printf("\n");
    for(i = 0; cadena[i] != '\0'; i++)
    {
        printf("%c", cadena[i]);
    }
    return cadena[60];

}

// FUNCION QUE REALIZA 5 ACCIONES DE LAS FUNCIONES ANTERIORES //
void todas(char cadena[60])
{
    int i;
    char original[60], original2[60], original3[60], original4[60];
    for (i = 0; cadena[i] != '\0'; i++)
    {
        original[i] = cadena[i];
        original2[i] = cadena[i];
        original3[i] = cadena[i];
        original4[i] = cadena[i];
    }
    noespacios(cadena);
    mayusculas(original);
    minusculas(original2);
    voltear(original3);
    capital(original4);

}
```

```
193
194  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
195  ////////////////////////////////////////FUNCIONES NUEVAS/////////////////////////////////////////////////////////
196  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
197
198  // FUNCION QUE VALIDA NUMEROS //
199  int valida_num(int ri, int rf)
200  {
201      int n;
202      char xnum[30];
203      printf("\nDame un numero entre el %d y %d: ", ri, rf);
204      scanf("%c", &xnum);
205      fflush(stdin);
206      gets(xnum);
207      n = atoi(xnum);
208      if (n > rf)
209      {
210          printf("\nNUMERO FUERA DE RANGO");
211          n = 70;
212      }
213
214      if (n < ri)
215      {
216          printf("\nNUMERO MUY PEQUEÑO PARA EL RANGO");
217          n = 30;
218      }
219
220      return n;
221  }
222
223  // FUNCION QUE EVITA NUMEROS REPETIDOS EN VECTOR //
224  bool no_repetir(int vect[],int n)
225  {
226      int i;
227      for (i = 0; i < 15; i++)
228      {
229          if (n == vect[i])
230          {
231              return true;
232          }
233      }
234      return false;
235  }
236
237  // FUNCION QUE LLENA VECTOR ALEATORIAMENTE //
238  void llenar_vect_aleatorio(int vect[], int m, int ri, int rf)
239  {
240      int rango, i, n;
241      rango = (rf - ri) + 1;
```

```
    int rango, i, n;
    rango = (rf - ri) + 1;
    srand(time(NULL));
    for(i = 0; i < m; i++)
    {
        do
        {
            n = (rand()%rango) + ri;
        } while (no_repetir(vect, n));
        vect[i] = n;
    }
}

// FUNCION QUE LLENA MATRIZ DE FORMA ALEATORIA //
void llenar_matriz(int matriz[4][4], int m, int ri, int rf)
{
    int rango, i, n, j;
    rango = (rf - ri) + 1;
    srand(time(NULL));
    for(j = 0; j < 4; j++)
    {
        for(i = 0; i < 4; i++)
        {
            do{
                n = (rand()%rango) + ri;
                } while (no_repetir_m(matriz, n));
        matriz[j][i] = n;
        }
    }
}

// FUNCION QUE IMPRIME LA MATRIZ //
void imprimir_mat(int matriz[4][4], int m, int n)
{
    printf("\nMATRIZ");
    int j, i;
    printf("\n");
    for (j = 0; j < m; j++)
    {
        printf("\n[");
        for(i = 0; i < n; i++)
        {
            printf("%d, ", matriz[j][i]);
        }
        printf("]");
    }
    printf("\n");

}
```

```
288 }
289
290 // FUNCION QUE IMPRIME VECTOR //
291 void imprimir_vect(int vect[], int m)
292 {
293     int i;
294     printf("\nVECTOR");
295     printf("\n[");
296     for (i = 0; i < m; i++)
297     {
298         printf("%d, ", vect[i]);
299     }
300     printf("]");
301     printf("\n");
302 }
303
304 // FUNCION QUE EVITA NUMEROS REPETIDOS EN MATRIZ //
305 bool no_repetir_m(int matriz[4][4],int n)
306 {
307     int i, j;
308     for (j = 0; j < 4; j++)
309     {
310         for (i = 0; i < 4; i++)
311         {
312             if (n == matriz[j][i])
313             {
314                 return true;
315             }
316         }
317     }
318     return false;
319 }
320
321 // FUNCION QUE ORDENA EL VECTOR //
322 void ordenar(int vect[], int m)
323 {
324     int temp, i, j;
325     for (i = 0; i < m - 1; i++)
326     {
327         for (j = i + 1; j < m; j++)
328         {
329             if (vect[j] < vect[i])
330             {
331                 temp = vect[i];
332                 vect[i] = vect[j];
333                 vect[j] = temp;
334             }
335         }
336     }
```

```
335             }
336         }
337 
338         printf("\nVECTOR ORDENADO");
339         printf("\n[");
340         for (i = 0; i < m; i++)
341         {
342             printf("%d, ", vect[i]);
343         }
344         printf("]");
345         printf("\n");
346     }
347 
348     // FUNCION PARA BUSQUEDA SECUENCIA //
349     int busq_sec(int vect[], int n, int num)
350     {
351         int i;
352         n = num;
353         for (i = 0; i < n; i++)
354         {
355             if(num == vect[i])
356             {
357                 return i;
358             }
359         }
360         return -1;
361     }
362 
363     // FUNCION PARA BUSCAR NUMERO //
364     void buscar (int vect[], int m)
365     {
366         int num, x, i;
367         num = valida_num(100, 200);
368         x = busq_sec(vect, M, num);
369         if (x != -1)
370         {
371             printf("\nSi existe");
372             printf("\n%d esta en el indice %d", num, x);
373         }
374         else
375         {
376             printf("\nNo existe");
377         }
378     }
```

PROGRAMA PRINCIPAL (.cpp)

HDCL_ACT9_01_432.cpp > ...

```cpp
//Hector Daniel Camacho Lopez     372239                                                    //
//Ensenada, Baja California a 20 de Septiembre del 2023                                     //
//Pograma que despliega un menu que trabaja con vectores y matrices usando una libreria personal //
//Nombre de la actividad: HDCL_ACT9_01_432                                                  //

#include "daniel.h"
#define M 15
#define N 4

int main()
{
    int decision, vect[M], m, matriz[4][4];
    char cadena[30];
    do{
        printf("\n          MENU            ");
        printf("\n1) Llenar un vector aleatoriamente con 15 numeros del 100 al 200 sin repetir");
        printf("\n2) LLenar matriz 4x4 con numeros aleatorios del 1 al 16 sin repetir");
        printf("\n3) Imprimir el vector de 15 numeros");
        printf("\n4) Imprimir la matriz de 4x4");
        printf("\n5) Ordenar el vector de 15 numeros");
        printf("\n6) Buscar un valor especifico en el vector de 15 numeros");
        printf("\n0) SALIR");
        printf("\nCual de las acciones anteriores deseas realizar? Ingresa el numero correspondiente: ");
        scanf("%d", &decision);
        switch(decision)
        {
            case 1:
                printf("\nHas escogido la opcion 1");
                llenar_vect_aleatorio(vect, M, 100, 200);
                break;

            case 2:
                printf("\nHas escogido la opcion 2");
                llenar_matriz(matriz, M+1, 0, 16);
                break;

            case 3:
                printf("\nHas escogido la opcion 3");
                imprimir_vect(vect, M);
                break;

            case 4:
                printf("\nHas escogido la opcion 4");
                imprimir_mat(matriz, M-11, N);
                break;

            case 5:
                printf("\nHas escogido la opcion 5");
                ordenar(vect, M);
```

```
        printf("\n5) Ordenar el vector de 15 numeros");
        printf("\n6) Buscar un valor especifico en el vector de 15 numeros");
        printf("\n0) SALIR");
        printf("\nCual de las acciones anteriores deseas realizar? Ingresa el numero correspondiente: ");
        scanf("%d", &decision);
        switch(decision)
        {
            case 1:
                printf("\nHas escogido la opcion 1");
                llenar_vect_aleatorio(vect, M, 100, 200);
                break;

            case 2:
                printf("\nHas escogido la opcion 2");
                llenar_matriz(matriz, M+1, 0, 16);
                break;

            case 3:
                printf("\nHas escogido la opcion 3");
                imprimir_vect(vect, M);
                break;

            case 4:
                printf("\nHas escogido la opcion 4");
                imprimir_mat(matriz, M-11, N);
                break;

            case 5:
                printf("\nHas escogido la opcion 5");
                ordenar(vect, M);
                break;

            case 6:
                printf("\nHas escogido la opcion 6");
                buscar (vect, M);
                break;

            case 0:
                printf("\nHAS DECIDIDO SALIR");
                break;

            default:
                printf("\nElige una de las opciones DENTRO del menu");

        }

    }while(decision != 0);
}
```