Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño F.I.A.D.

Campus Ensenada

Ramón Mejía Martínez 372099

Grupo: 932

Actividad 9

Programación Estructurada

Ensenada, Baja California a 30 de septiembre de 2023.

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Ramón Mejía Martínez

Matrícula: 372099

Maestro: Pedro Núñez Yépiz

Actividad No. : 9

Tema - Unidad : Librerías en c, métodos de ordenación y búsqueda.

Ensenada Baja California a 8 de octubre del 2023.

1. Introducción

En esta actividad se usarán el método de búsqueda secuencial que recorre todo el arreglo para encontrar un elemento en él, el método de la burbuja para ordenar elementos y el uso de librerías propias para el desarrollo de programas.

2. Competencia

En este reporte se aprenderá sobre algunos algoritmos de ordenación y búsqueda para trabajar con arreglos, el uso y creación de librerías para poder controlar todos los procesos que realiza nuestro código. Los algoritmos de ordenación son importantes para organizar datos en un orden específico, mientras que los algoritmos de búsqueda permiten encontrar un elemento particular en un conjunto de datos.

3. Fundamentos

Búsqueda secuencial:

Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta encontrarlo.

Ejemplo:

- Si buscamos el numero 8 en el siguiente arreglo para comprobar que existe en el arreglo:
  8==1? Si no es igual avanza a la siguiente casilla
  8==3? Va avanzando hasta llegar al valor
  . . .
  8==8? Entonces encontramos la posición la cual es 3 ya que el arreglo inicia desde 0.

1	3	10	8	7	93	2	5	34	12

Método de la burbuja mejorado:

El método de la burbuja consiste en ir comparando las posiciones consecutivas para comprobar si alguno es mayor, va tomando el numero mayor y lo lleva hasta el final de arreglo.

La única diferencia entre el método de la burbuja mejorado y el normal es que se va comprobando si se hizo algún intercambio de valores en el arreglo, si no se hizo termina el método y así nos ahorramos varias comparaciones si el arreglo ya esta ordenado.

Librerías en C.

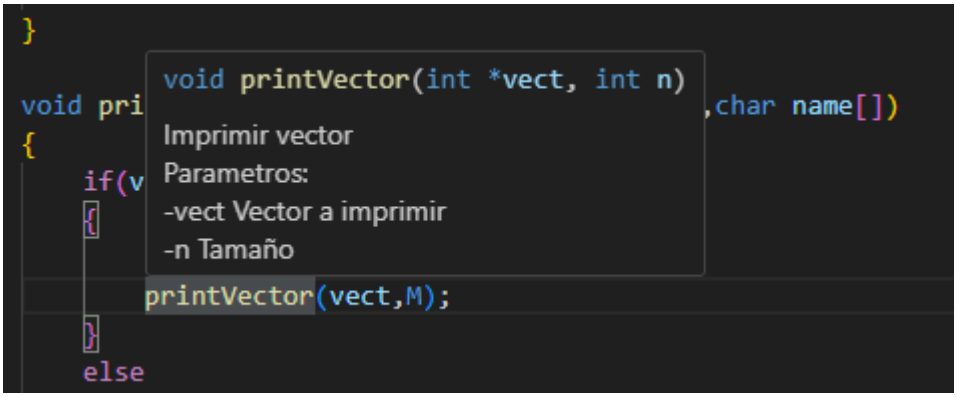
Las librerías en c son archivos o bloques de código con funciones especiales que podemos incluir en nuestro código importando sus funcionalidades.

Para la creación de nuestra propia librería podemos crear un archivo.h que contendrá todas la funciones que deseemos, este archivo puede contener los prototipos de las funciones e incluir los cuerpos en otro archivo, también puede contener tanto prototipos como cuerpos. Para importar estas librerías en nuestros archivos.c se hace de la siguiente manera:

#include “(ruta de la librería)”

Usando la palabra reservada #include con la que se agregan otras librerías, entre comillas escribimos la ruta de nuestra librería, ahora tendremos todas las funciones existentes en el archivo.h para nuestro archivo.c.

Para Visual Studio Code podemos escribir una descripción de la función con comentarios en el archivo.h y se mostrara en nuestro archivo.c para mejor entendimiento del código.



4. Procedimiento

ACTIVIDAD 9

Realiza programa en C utilizando librería propia, el programa deberá tener el siguiente menú.

MENÚ
1.- LLENAR VECTOR
2.- LLENAR MATRIZ
3.- IMPRIMIR VECTOR
4.- IMPRIMIR MATRIZ
5.- ORDENAR VECTOR
6.- BUSCAR VALOR EN VECTOR
0.- SALIR

NOTA: El programa deberá repetirse cuantas veces lo desee el usuario, Validado el menú con la función vali_num

I N S T R U C C I O N E S

1.- LLENAR VECTOR .- Llenar vector con 15 números, los números generados aleatoriamente, los números entre el rango de 100 al 200 **(no repetidos)**
2.- LLENAR MATRIZ .- Llenar la matriz de 4x4 con con números generados aleatoriamente, números entre el rango de 1 al 16 **(no repetidos)**
3.- IMPRIMIR VECTOR .- Imprime el vector que se envíe, donde la función recibe como parámetro el vector,tamaño, nombre del vector.
4.- IMPRIMIR MATRIZ.- Imprime la matriz sin importar el tamaño de la matriz recibiendo como parámetros la matriz, la cantidad de renglones y columnas, así como nombre que se le dará a la matriz
5.- ORDENAR VECTOR.- Usar función que ordene el vector por el método de ordenación de la **Burbuja mejorada**.
6.- BUSCAR VALOR EN VECTOR.- Buscar un valor en el vector usando el método de **búsqueda secuencial**.
0.- SALIR

5. Resultados y conclusiones

Resultados:

Opción 1 y 2:

- Usamos directamente dos funciones de nuestra librería.

```
case 1:
    v1=randVector(vect,M,100,200);
    printf("\nSe lleno el vector aletorio\n");
    break;
case 2:
    m1=randMatriz(matriz,NM,NM,1,16);
    printf("\nSe lleno la matriz aleatoria\n");
    break;
```

Función randVector:

- Recibe:
  - Vector a llenar.
  - Tamaño del vector.
  - Rango inicial y final para valor aleatorio.
- Recorremos todas las posiciones del vector con un for.
- Dentro de un dowhile generamos un numero aleatorio, en la condición validamos si ya esta en el vector, si lo esta vuelve a generarlo, si no sale del dowhile.
- Asignamos el numero al vector.
- Regresa:
  - 1, representa que se lleno el vector.

```
int randVector(int vect[],int n,int ri,int rf)
{
    int i,num,range;
    srand(time(NULL));
    range=(rf-ri)+1;
    for(i=0;i<n;i++)
    {
        do
        {
            num=(rand()%range)+ri;
        }while(searchVector(vect,n,num)!=-1);
        vect[i]=num;
    }
    return 1;
}
```

Función randMatriz:

- Recibe:
  - Matriz a llenar.
  - Numero de renglones.
  - Numero de columnas.
  - Rango inicial y final para valor aleatorio.
- Usamos randVector para generar el número de valores aleatorios.
- Llenamos la matriz con los valores del vector.

```
int randMatriz(int matriz[][NM],int r,int c,int ri,int rf)
{
    int i,j,k;
    int n;
    n=r*c;
    int vect[n];
    randVector(vect,n,ri,rf);
    k=0;
    for(i=0;i<r;i++)
    {
        for(j=0;j<r;j++)
        {
            matriz[i][j]=vect[k];
            k++;
        }
    }
    return 1;
}
```

Opción 3:

Llamamos a una función que recibe el vector a imprimir, el tamaño, la bandera para confirmar si esta lleno o no y el nombre del vector.

- Si la bandera es 1 entonces imprime el vector llamando a una función de nuestra librería.
- Si la bandera no es 1 entonces indica que esta vacía y vuelve al menú.

```
void printVectorACT9(int vect[],int n,int v1,char name[])
{
    if(v1==1)
    {
        printf("\n%s\n\n",name);
        printVector(vect,n);
    }
    else
    {
        printf("\n%s esta vacio",name);
    }
    printf("\n");
}
```

```
void printVector(int vect[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d ",vect[i]);
    }
}
```

Opción 4:

Usamos el mismo concepto que usamos en la opción 3.

```
void printMatrizACT9(int matriz[][NM],int r, int c,int m1,char name[])
{
    if(m1==1)
    {
        printf("\n%s\n\n",name);
        printMatriz(matriz,r,c);
    }
    else
    {
        printf("\n%s esta vacia\n",name);
    }
}
```

```
void printMatriz(int matriz[][NM],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%4d ",matriz[i][j]);
        }
        printf("\n");
    }
}
```

Opción 5:

- Si el vector esta lleno llamamos a la función del método de la burbuja mejorado.

```
case 5:
    if(v1==1)
    {
        improvedBubbleSort(vect,M);
    }
    else
    {
        printf("\nEl vector esta vacio\n");
    }
    break;
```

```
void improvedBubbleSort(int vect[],int n)
{
    int i,j;
    int temp,swap;
    for(i=0;i<n-1;i++)
    {
        swap=0;
        for(j=0;j<n-i-1;j++)
        {
            if(vect[j]>vect[j+1])
            {
                temp=vect[j];
                vect[j]=vect[j+1];
                vect[j+1]=temp;
                swap=1;
            }
        }
        if (swap==0)
        {
            i=n;
        }
    }
}
```

Opción 6:

- Si la bandera del vector es 1 entonces preguntamos el número a buscar.
  - Con la función searchVector de nuestra librería buscamos si está el número en el vector.
  - Si lo encontró indicamos la posición, si no lo encontró indicamos eso.
- Si la bandera del vector no es 1, entonces esta vacío, se lo hacemos saber al usuario y regresa al menú.

```
void searchVectorACT9(int vect[],int n,int v1)
{
    int num,pos;
    if(v1==1)
    {
        num=validNum("Ingresa el numero a buscar: ",100,200);
        pos=searchVector(vect,n,num);
        if(pos!=-1)
        {
            printf("\nEl valor %d esta en la posicion %d del vector\n",num,pos);
        }
        else
        {
            printf("\nEl valor %d no esta en el vector\n",num);
        }
    }
    else
    {
        printf("\nEl vector esta vacio\n");
    }
}
```

Usando la búsqueda secuencial buscamos un valor en el vector.

```
int searchVector(int vect[],int n,int num)
{
    int i;
    for(i=0;i<n;i++)
    {
        if(vect[i]==num)
        {
            return i;
        }
    }
    return -1;
}
```

Conclusión:

Esta actividad nos ayuda a recopilar mas funciones útiles como lo son los métodos de ordenamiento y búsqueda, el método de la burbuja y el de búsqueda secuencial para incluirlos en nuestra librería y poder usarlos en programas futuros.

6. Anexos

PDF:

https://drive.google.com/file/d/1vuwTJNsUrEKJoAlQKAX3N14LEesJTDwk/view?usp=sharing

Repositorio:

https://github.com/Muners24/Programacion-Estructurada

7. REFERENCIAS

Diseño de algoritmos y su codificación en lenguaje C

Corona, M.A. y Ancona, M.A. (2011)..

España: McGraw-Hill.

ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C

:Pearson Educación.Sznajdleder, P. A. (2017)..

Buenos Aires,Argentina: Alfaomega

Como programar en C/C++

H.M. Deitel/ P.J. Deitel

Segunda edición

Editorial: Prentice Hall.

ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos

Joyanes, L. y Zahonero, I. (2001)..

España:McGraw-Hill.

ISBN: 8448130138

De Computadores, P., & Olave, T. A. (s/f). *Algoritmos de Búsqueda y Ordenamiento*.

Utfsm.cl. Recuperado el 9 de octubre de 2023, de https://www.inf.utfsm.cl/~noell/IWI-131-p1/Tema8b.pdf