# Introducción a funciones:

1. Definir las siguientes funciones:
2. max(num1,num2); //esta función se le pasan dos números enteros por parámetro y nos devuelve el mayor de ellos.
3. min(num1,num2); //se le pasan dos números enteros por parámetro y devuelve el menor de ellos.
4. esPar(num); //se pasa un numero por parámetro y devuelve True en caso de ser par
5. El arreglo a utilizar en esta práctica será de tipo Int con 100 elementos NO repetidos, el arreglo podrá contener una cantidad de números =< 100, siempre y cuando sean solo positivos, los campos libres que no tengan números contendrán el número -1, esto indica que esta ese lugar esta vacío.

Ejemplo de arrays validos:

[12,32,67,1,43,74,41,21,6,-1,-1,-1,-1,-1,.…,-1]

[34,16,22,99,130,1,223,-1,-1,-1,-1,-1,-1,.…,-1]

[9,5,7,1,3,2,4,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,…,-1]

Para comenzar, la forma de crear el array será la siguiente:

Int nombre\_del\_array[100];

cargarArray(nombre\_del\_array);

1. Definir la función cargarArray(array[]); la cual recibe por parámetro un array, dentro de la función se pedirá ingresar por teclado a los sumo 100 números positivos no repetidos, estos se irán cargando dentro del array, cuando se ingrese el número -1 se finaliza la carga de números, si quedan espacios vacíos en el array hay que cargar esos espacios con el número -1 para indicar que están vacíos.
2. Escriba un programa en el que se hace uso de la función ante descripta y luego imprima el contendió de todo el array en pantalla.
3. Defina la función tamanio(array[]); recibirá por parámetro un array igual al descripto anteriormente, la función lo recorrerá buscando el primer casillero con el número -1 y contabilizando la cantidad de elementos que tiene el array, la función debe retorna la cantidad de elemento que tiene el array.

Ej: array[15,32,22,1,92,2,1,111,-1,-1,-1,-1….,-1]

cant\_elementos = tamanio(array);

cant\_elementos >> 8 🡨 cantidad de elementos en el array

1. Defina la función buscar(array[], num); recibe por parámetro el arreglo y un elemento que está dentro del arreglo y se desea saber la posición del mismo. la función retorna la posición del elemento buscado en el arreglo en caso de no encontrarse dentro del arreglo, retornara -1.
2. Haciendo uso de las funciones anteriormente definidas e implementando nuevas cuando crea necesario hacer un programa con un menú de opciones que realice las siguientes operaciones:
3. Crea un array como el definido anteriormente y lo imprime en pantalla.
4. Calcula el valor promedio de los elementos del array.
5. Imprime la cantidad de elementos pares e impares que tiene el array.
6. Ordena un array de menor a mayor
7. Agregue al programa anterior una opción para eliminar un elemento del array, tener en cuenta que luego de eliminar el elemento (si es que se encuentra en el array) deberá compactarlo, esto significa que si el elemento eliminado se encuentra en una posición intermedia, entre los elementos del array, deberá mover todos los elementos del array, que se encuentras luego de ese, una posición hacia atrás.

Ej:

array=[32,22,40,51,17,2,7,-1,-1,-1,-1,-1….,-1]

eliminar(array,40);

array >> [32,22,-1,51,17,2,7,-1,-1,-1,-1,-1….,-1] 🡨donde estaba el 40 se puso un -1

compactar(array);

array >> [32,22,51,17,2,7,-1,-1,-1,-1,-1,-1….-1]🡨 luego de compactar el array

1. Cree otro array del mismo tipo que el anterior y cárguelo con elementos, luego genere un tercer array que va ser la suma del primer array y el segundo, a su vez este 3 array tiene que quedar ordenado de menor a mayor.

Ej: array1=[32,22,40,51,17,2,7,-1,-1,-1,-1,-1….,-1]

array2=[18,21,33,99,-1,-1,-1,-1…..,-1]

array3>>[2,7,17,18,21,22,32,33,40,51,99]

Aclaración: tenga en cuenta el tamaño de los arrays y la posibilidad de que allá elementos repetidos.