	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación salas A y B

Profesor : Edgar Tista García

Asignatura: EDA 2

Grupo: 4

No. Práctica : 3

Integrante: Chagoya Gonzalez Leonardo

No. Equipo de Cómputo: N/A

No. de Lista o Brigada: N/A

Semestre: 2022-2

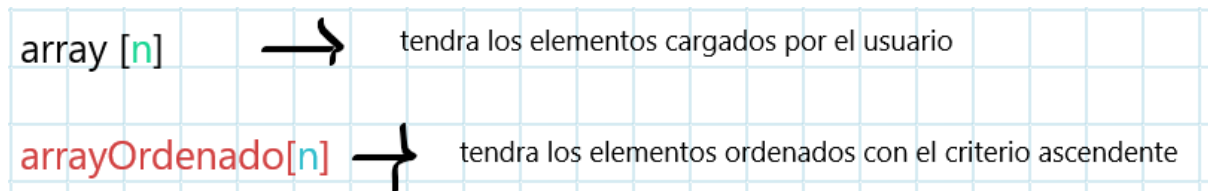
Fecha de entrega: 02/03/2022

Observaciones:

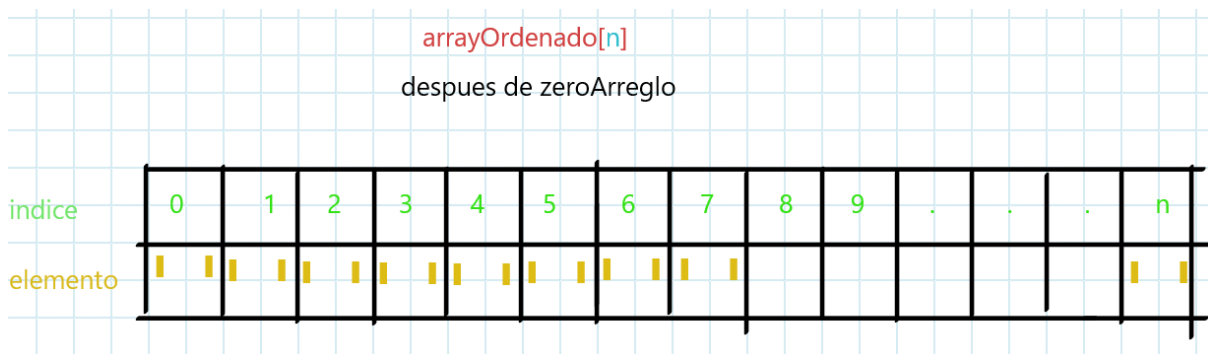
CALIFICACIÓN: _____

Ejercicio 1. Counting Sort

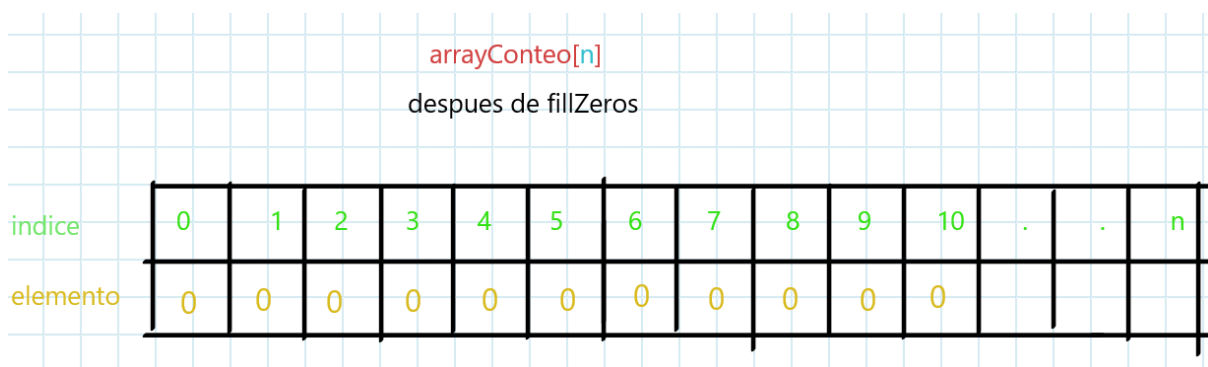
La implementación del algoritmo fue realizada en el lenguaje c, definiendo inicialmente 2 arreglos de n elementos, uno contendrá la carga de elementos por parte del usuario - *array* - mientras que el segundo arreglo tendrá dichos los elementos ordenados de forma Ascendente - *arrayOrdenado* - , como los elementos serán caracteres que van de a-j el criterio ascendente se considera como: a,b,c,d.....



De forma auxiliar inicializamos el *arrayOrdenados* con caracteres vacíos mediante la función *ZeroArreglo*, ya que cuando se crea un arreglo y no se inicializa este tiene información “basura” en cada posición.



Posteriormente se llama a la función *countingSort* pasando como parámetros ambos arreglos anteriormente mencionados junto con el número de elementos, dentro de la función se crea un arreglo del tamaño del rango de elementos llamado *arrayConteo*, en este ejercicio nuestro rango de elementos es 10 ya que solo aceptaremos letras desde *a* hasta *j*, dicho arreglo será inicializado en 0 para posteriormente realizar la cuenta de los elementos en *array*.



Dentro del conteo es donde se me presentó la primera dificultad, ya que cada elemento de *array* debe de estar asociado con un índice de *arrayConteo* para aumentar de valor el contenido del índice conforme se encuentra un elemento. Para solucionar esto me apoye en la tabla de código ASCII en donde a cada carácter se le asigna un valor y restando el valor del carácter “a” en ASCII menos el carácter en turno de *array* se logra obtener la posición en *arrayConteo* para incrementar en 1.

Letra	a	b	c	d	e	f	g	h	i	j
Valor Código ASCII	97	98	99	100	101	102	103	104	105	106
índice <u>arrayConteo</u>	0	1	2	3	4	5	6	7	8	9

Este pensamiento funciona para todas las letras minúsculas , teniendo que aumentar el rango, sin embargo el ejercicio pedía hasta la letra j.

Una vez realizado el conteo se procede a realizar la suma de $i + (i-1)$ en *arrayConteo* que es el elemento en turno más el elemento anterior.

Hasta este punto se ha seguido la implementación del algoritmo visto en clase para listas de 1 hasta n elementos, sin embargo en los lenguajes de programación un arreglo va de 0 hasta n-1 en donde 0 es el primer elemento del arreglo, es decir hay un pequeño desfase en las posiciones que deben de ir los elementos para ello se modifica *arrayConteo* quitándole a cada uno de sus elementos -1, en un principio no había notado este punto y al momento de imprimir el arreglo ordenado en el primer elemento aparecía una cadena vacía , como primera solución propuse inicializar el *arrayConteo* en -1 sin embargo al realizar el conteo y la suma había error lógico, ya que el número 0 en *arrayConteo* corresponde a que existe un elemento en dicha posición del arreglo pero 0 aritméticamente no representa un incremento , fue por esto que al final se decidió realizar el acomodo de esta variación una vez hecho el conteo y la suma.

Posteriormente mediante la función *couting* se recorre *array* del final hasta el inicio colocando el elemento en turno en *arrayOrdenado* conforme *arrayConteo* disminuye según el elemento.

Pruebas de Ejecución

Carga del arreglo y conteo

```
-----EJERCICIO 1-----
Carga del arreglo
Ingrese un caracter: a
Ingrese un caracter: b
Ingrese un caracter: c
Ingrese un caracter: d
Ingrese un caracter: e
Ingrese un caracter: f
Ingrese un caracter: g
Ingrese un caracter: h
Ingrese un caracter: i
Ingrese un caracter: j
Ingrese un caracter: j
Ingrese un caracter: a
Ingrese un caracter: d
Ingrese un caracter: e
Ingrese un caracter: a
Ingrese un caracter: a
Ingrese un caracter: b
Ingrese un caracter: f
Ingrese un caracter: g
Ingrese un caracter: c
Carga completada
```

```

EL numero de apariciones de cada elemento
a      b      c      d      e      f      g      h      i      j
4      2      2      2      2      2      2      1      1      2

Una vez realizada la suma
a      b      c      d      e      f      g      h      i      j
4      6      8      10     12     14     16     17     18     20

Acomodando para que coincida con los indices
a      b      c      d      e      f      g      h      i      j
3      5      7      9      11     13     15     16     17     19
```

Ingresando los elementos en *arrayOrdenado*

```
La letra es: c
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
                                c

El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
3   5   6   9   11  13  15  16  17  19

La letra es: g
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
                                g

El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
3   5   6   9   11  13  14  16  17  19

La letra es: f
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
                                f
                                g

El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
3   5   6   9   11  12  14  16  17  19

La letra es: b
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
                                b
                                c
                                f
                                g

El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
3   4   6   9   11  12  14  16  17  19
```

```
La letra es: a
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
a   a   b   c   f   g
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
1   4   6   9   11  12  14  16  17  19

La letra es: e
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
a   a   b   c   e   f   g
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
1   4   6   9   10  12  14  16  17  19

La letra es: d
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
a   a   b   c   d   e   f   g   h   i   j
1   4   6   8   10  12  14  16  17  19

La letra es: a
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
a   a   a   b   c   d   e   f   g
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  14  16  17  19
```

```

La letra es: j
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  14  16  17  18

La letra es: j
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  14  16  17  17

La letra es: i
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  14  16  16  17

La letra es: h
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  14  15  16  17

La letra es: g
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  12  13  15  16  17

```

```

La letra es: f
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   10  11  13  15  16  17

La letra es: e
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   8   9   11  13  15  16  17

La letra es: d
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   d   e   f   g   h   i   j   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   6   7   9   11  13  15  16  17

La letra es: c
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   c   c   d   d   e   e   f   f   g   g   h   i   j   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   4   5   7   9   11  13  15  16  17

La letra es: b
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
    a   a   a   b   b   c   c   d   d   e   e   f   f   g   g   h   i   j   j
El arreglo contador es:
a   b   c   d   e   f   g   h   i   j
0   3   5   7   9   11  13  15  16  17

```

```

La letra es: a
Ingresando la letra:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  a   a   a   a   b   b   c   c   d   d   e   e   f   f   g   g   h   i   j   j
El arreglo contador es:
a     b     c     d     e     f     g     h     i     j
-1    3     5     7     9    11    13    15    16    17

```

```

El arreglo ordenado es:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  a   a   a   a   b   b   c   c   d   d   e   e   f   f   g   g   h   i   j   j

```

Ejercicio 2. RadixSort

Para la implementación de este algoritmo se utilizó el lenguaje C, teniendo un arreglo de n elementos al cual el usuario ingresara dichos elementos mediante la función *cargaArreglo*

Una vez hecha la carga del arreglo se procederá a ejecutar la función *radixSort* en la cual se realizará el análisis conforme a las unidades, decenas, centenas o miles de los números del arreglo para realizar dicho análisis se ocupa la función *sort* a la cual se le pasa como parámetro el arreglo, la cantidad de elementos y una función que corresponderá a la parte del número a analizar.

```

//analizando las unidades           //analizando las decenas
printf("\nIteracion 1:\n");         printf("\nIteracion 2:\n");
sort(array,n,unidades);             sort(array,n,decenas);
imprimirArreglo(array,n);           imprimirArreglo(array,n);

```

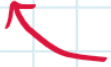
Dentro de la función *sort* se recorre el arreglo analizando cuántas unidades o decenas o centenas o miles tiene cada elemento, según sea el caso se crea un nodo en donde el campo *info* tendrá el elemento del arreglo en turno y lo inserta en la estructura cola correspondiente

Al final se imprime como están las estructuras colas en dicho momento y quita los elementos de dichas estructuras empezando desde el inicio.

La dificultad de implementar este algoritmo es que en principio cada pasada al arreglo se tiene que analizar unidades conforme a este análisis se colocará el elemento en turno en una cola al finalizar la pasada se extraerán los elementos de las estructuras colas y se repetirá pero ahora con decenas lo que implicaría repetir el código dentro de la función *sort* con la excepción que ahora el cálculo para el análisis tendría que ser con decenas, posteriormente lo mismo con centenas y miles.

Fue por ello que en *radixSort* se decidió pasar como parámetro una función, dicha función indicará que parte del número se está analizando y ejecutará el algoritmo de analizar los números y colocarlos en la estructura correspondiente para luego aplicarles los pop correspondientes.

```
void sort(int *array, int n, int (*)(int elem) )
```



realizara el calculo de unidades, decenas, centenas o miles dependiendo sea el caso

Pruebas de Ejecución

Carga del arreglo

```
-----EJERCICIO 2-----
Carga del arreglo
Ingrese un numero max 4 digitos (digitos 1-4): 1234
Ingrese un numero max 4 digitos (digitos 1-4): 4321
Ingrese un numero max 4 digitos (digitos 1-4): 2321
Ingrese un numero max 4 digitos (digitos 1-4): 4343
Ingrese un numero max 4 digitos (digitos 1-4): 1441
Ingrese un numero max 4 digitos (digitos 1-4): 2413
Ingrese un numero max 4 digitos (digitos 1-4): 1412
Ingrese un numero max 4 digitos (digitos 1-4): 2222
Ingrese un numero max 4 digitos (digitos 1-4): 3333
Ingrese un numero max 4 digitos (digitos 1-4): 3141
Ingrese un numero max 4 digitos (digitos 1-4): 1314
Ingrese un numero max 4 digitos (digitos 1-4): 3123
Ingrese un numero max 4 digitos (digitos 1-4): 3432
Ingrese un numero max 4 digitos (digitos 1-4): 2232
Ingrese un numero max 4 digitos (digitos 1-4): 2211
Carga completada
El arreglo original es:
1234 4321 2321 4343 1441 2413 1412 2222 3333 3141 1314 3123 3432 2232 2211
```


Muestra de iteraciones y arreglo Ordenado

```
Iteracion 1:
Cola 1: [4321] [2321] [1441] [3141] [2211]
Cola 2: [1412] [2222] [3432] [2232]
Cola 3: [4343] [2413] [3333] [3123]
Cola 4: [1234] [1314]
4321 2321 1441 3141 2211 1412 2222 3432 2232 4343 2413 3333 3123 1234 1314

Iteracion 2:
Cola 1: [2211] [1412] [2413] [1314]
Cola 2: [4321] [2321] [2222] [3123]
Cola 3: [3432] [2232] [3333] [1234]
Cola 4: [1441] [3141] [4343]
2211 1412 2413 1314 4321 2321 2222 3123 3432 2232 3333 1234 1441 3141 4343

Iteracion 3:
Cola 1: [3123] [3141]
Cola 2: [2211] [2222] [2232] [1234]
Cola 3: [1314] [4321] [2321] [3333] [4343]
Cola 4: [1412] [2413] [3432] [1441]
3123 3141 2211 2222 2232 1234 1314 4321 2321 3333 4343 1412 2413 3432 1441

Iteracion 4:
Cola 1: [1234] [1314] [1412] [1441]
Cola 2: [2211] [2222] [2232] [2321] [2413]
Cola 3: [3123] [3141] [3333] [3432]
Cola 4: [4321] [4343]
1234 1314 1412 1441 2211 2222 2232 2321 2413 3123 3141 3333 3432 4321 4343

La Lista ordenada es:
1234 1314 1412 1441 2211 2222 2232 2321 2413 3123 3141 3333 3432 4321 4343
```

Ejercicio 0

Para este ejercicio se agregaron las bibliotecas con sus respectivas funciones para cada uno de los algoritmos a un nuevo proyecto y el menú se encuentra en la función main, teniendo que declarar los arreglos para cada uno de los algoritmos sin saber si se van a utilizar, al ser casos particulares donde uno es un arreglo de caracteres mientras que otro es un arreglo de enteros no pude ocupar solo un arreglo, aunque otra opción sería declarar dichos arreglos dentro de las funciones de sus algoritmos, sin embargo prefiero que el arreglo ordenado independientemente de que algoritmo se ocupe se quede dentro de la función principal.

Conclusiones

Para los algoritmos de counting Sort y de radixSort vistos en clase no presente mayor dificultad para entender el funcionamiento de cada uno en comparación de quickSort o MergeSort, teniendo dichos funcionamientos presentes las dificultades presentadas fue en el momento de pasarlo al lenguaje de programación C es decir la implementación.

Para counting Sort la dificultad estuvo en asociar los elementos del arreglo a ordenar con el índice del arreglo conteo, sin embargo fue solucionado haciendo uso de la tabla de código ASCII y una resta, así como también en acomodar el desfase del conteo ya que nuestro primer elemento en un arreglo comenzará desde la posición 0.

Para radixSort la dificultad estuvo en analizar cómo hacer de mejor manera cada iteración para analizar unidades, decenas, centenas o millares. Sin embargo después de realizar los respectivos puntos se logró implementar ambos algoritmos cumpliendo el 100% de los ejercicios en esta práctica y con ello cumpliendo los objetivos.

Decidí realizar esta práctica en C ya que en comparación de Java me siento más familiarizado con C y en caso de tener algún error de lógica en la implementación de los algoritmos sentí que era más fácil para mi solucionarlos en C que en Java, sin embargo para el ejercicio de RadixSort por la forma en que Java trabaja las estructuras colas y con el paradigma orientado a objetos pudo haber facilitado la implementación de dicho algoritmo.