

Benemérita Universidad Autónoma de Puebla  
Facultad de Ciencias de la Computación

# PROGRAMACIÓN I

UNIDAD 2: METODOS DE ORDENAMIENTO Y BÚSQUEDA



Docente:

Prof.<sup>a</sup> Erika Bonfil Barragán



- EQUIPO 8

Jesús Huerta Aguilar		202041509
Javier De La Luz Ruiz		202033810
Ernesto Flores Cesáreo		202066335

Fecha de elaboración:

30/09/2021

NRC: 18438

Sección: 007

SEGUNDO SEMESTRE

Puebla, Pue.

Fecha de entrega: 01/10/2021

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

1. Implemente un programa en lenguaje C que programe los algoritmos de ordenamiento:

- a) Burbuja
- b) Inserción directa
- c) Selección directa

Mediante un menú se debe seleccionar el algoritmo para ordenar el arreglo.

Al seleccionar un algoritmo se pedirá el tamaño del arreglo y el arreglo a ordenar.

Se mostrará el resultado del arreglo ordenado.

Solo saldrá del programa con la opción de salida.

CODIGO:

```
1. //Jesús Huerta Aguilar, Javier de La Luz Ruiz, Ernesto Flores Cesareo
2. //Programación I - "Programa: Ordenamiento burbuja,
   inserción y selección"
3. #include <conio.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. //PROTOTIPOS
7. void menu(int *);
8. void nom(int);
9. void size(int *,int);
10. void lect(int [],int,int);
11. void impr(int [],int);
12. void burbu(int [],int);
13. void indi(int [],int);
14. void direct(int [],int);
15. void casos(int ,int [],int);
16. void salida(char *,int);
17. //PRINCIPAL
18. int main(){
19.     int *a,tam,op;
20.     char des;
21.     do{
22.         //MENU
23.         menu(&op);
24.         //TAMAÑO
25.         if (op != 4){
26.             //nom(op);
27.             size(&tam,op);
28.             a = (int *)calloc(tam,sizeof(int));
29.             //LECTURA
30.             lect(a,tam,op);
31.             //CASOS
32.             casos(op,a,tam);
33.         }
34.         salida(&des,op);
35.     } while (des == 's' || des == 'S');
36.     getch();
37.     return 0;
38. }
39. //MENU
40. void menu(int *op){
```

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

```
41.     int error;
42.     do{
43.         error = 0;
44.         printf("\t\t- - - ALGORITMOS DE ORDENAMIENTO - - -\n");
45.         printf("\n\t\t\t\t - || MENU || -");
46.         printf("\n\nSeleccione alguno de los siguientes m%ctodos de orden
amiento:",130);
47.         printf("\n\n\t[1]: Burbuja");
48.         printf("\n\t[2]: Inserci%cn directa",162);
49.         printf("\n\t[3]: Selecci%cn directa",162);
50.         printf("\n\t[4]: SALIR");
51.         printf("\n\n///// ");
52.         scanf("%d",op);
53.         if (*op < 1 || *op > 4){
54.             printf("\n[!] ERROR: Valores incorrectos [!]);
55.             error = 1;
56.             getch();
57.             system("cls");
58.         }
59.     } while (error == 1);
60.     system("cls");
61. }
62. //NOMBRE ACTUAL
63. void nom(int op){
64.     printf("\t\t- - - ALGORITMOS DE ORDENAMIENTO - - -\n");
65.     printf("\n///// M%ctODO DE ORDENAMIENTO: ",144);
66.     switch (op){
67.     case 1:
68.         printf("BURBUJA\n\n");
69.         break;
70.     case 2:
71.         printf("INSERCI%cN DIRECTA\n\n",224);
72.         break;
73.     case 3:
74.         printf("SELECCI%cN DIRECTA\n\n",224);
75.         break;
76.     }
77. }
78. //TAMAÑO
79. void size(int *tam,int op){
80.     int error;
81.     do{
82.         nom(op);
83.         error = 0;
84.         printf("Indique el tama%co del arreglo: ",164);
85.         scanf("%d",tam);
86.         if (*tam <= 1){
87.             printf("\n[!] ERROR: Valores incorrectos [!]);
88.             error = 1;
89.             getch();
90.             system("cls");
91.         }
92.     } while (error == 1);
93.     printf("\n");
94. }
```

```
95. //LECTURA
96. void lect(int arreg[], int tam, int op){
97.     int i;
98.     system("cls");
99.     nom(op);
100.     printf("Arreglo tama%co: %d\n\n", 164, tam);
101.     printf("Ingrese los valores del arreglo:\n\n");
102.     for (i = 0; i < tam; i++){
103.         printf("\t[%02d] %c ", i+1, 26);
104.         scanf("%d", &arreg[i]);
105.     }
106. }
107. //IMPRECION
108. void impr(int arreg[], int tam){
109.     int i;
110.     for (i = 0; i < tam; i++){
111.         printf("\n\t[%02d] %c %d", i+1, 26, arreg[i]);
112.     }
113. }
114. //BURBUJA
115. void burbu(int arreg[], int tam){
116.     int i, j, aux;
117.     for (i = 0; i < tam; i++){
118.         for (j = i + 1; j < tam; j++){
119.             if (arreg[j] < arreg[i]){
120.                 aux = arreg[i];
121.                 arreg[i] = arreg[j];
122.                 arreg[j] = aux;
123.             }
124.         }
125.     }
126.     impr(arreg, tam);
127. }
128. //INSERCIÓN DIRECTA
129. void indi(int arreg[], int tam){
130.     int i, act, aux;
131.     for (i = 0; i < tam; i++){
132.         act = i;
133.         while (act > 0 && arreg[act] < arreg[act - 1]){
134.             aux = arreg[act - 1];
135.             arreg[act - 1] = arreg[act];
136.             arreg[act] = aux;
137.             act = act - 1;
138.         }
139.     }
140.     impr(arreg, tam);
141. }
142. //SELECCIÓN DIRECTA
143. void direct(int arreg[], int tam){
144.     int i, j, min, aux;
145.     for (i = 0; i < tam-1; i++){
146.         min = i;
147.         for (j = i + 1; j < tam; j++){
148.             if (arreg[j] < arreg[min]){
149.                 min = j;
```

```
150.         }
151.     }
152.     aux = arreg[i];
153.     arreg[i] = arreg[min];
154.     arreg[min] = aux;
155. }
156. impr(arreg,tam);
157. }
158. //CASOS
159. void casos(int op,int arreg[],int tam){
160.     printf("\n///// ARREGLO ORDENADO:\n");
161.     switch (op){
162.     case 1:
163.         burbu(arreg,tam);
164.         break;
165.     case 2:
166.         indi(arreg,tam);
167.         break;
168.     case 3:
169.         direct(arreg,tam);
170.         break;
171.     }
172. }
173. //SALIDA
174. void salida(char *des,int op){
175.     int error;
176.     do{
177.         error = 0;
178.         if (op == 4){
179.             *des = 'N';
180.         }
181.         else{
182.             printf("\n\n%cDesea continuar? (S/N) >> ",168);
183.             scanf("%s",des);
184.         }
185.         if (*des == 's' || *des == 'S'){
186.             system("cls");
187.         }
188.         else{
189.             if (*des == 'n' || *des == 'N'){
190.                 system("cls");
191.                 printf("\n\n\tGRACIAS POR USAR NUESTROS SERVICIOS")
192.                 ;
193.                 printf("\n\n\t\tEquipo 8\t-\tBUAP");
194.             }
195.             else{
196.                 printf("\n[!] ERROR: Ingresa una opci%cn valida [!]",162);
197.                 error = 1;
198.             }
199.         } while (error == 1);
200.     }
```

EJECUCIÓN:

```
a:\Principal\Escritorio\Problemario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

- || MENU || -

Seleccione alguno de los siguientes métodos de ordenamiento:

[1]: Burbuja
[2]: Inserción directa
[3]: Selección directa
[4]: SALIR

///// 1
```

```
a:\Principal\Escritorio\Problemario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

///// MÉTODO DE ORDENAMIENTO: BURBUJA

Arreglo tamaño: 5

Ingrese los valores del arreglo:

[01] → 1
[02] → 5
[03] → 2
[04] → 4
[05] → 3

///// ARREGLO ORDENADO:

[01] → 1
[02] → 2
[03] → 3
[04] → 4
[05] → 5

¿Desea continuar? (S/N) >> s
```

```
a:\Principal\Escritorio\Probleuario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

- || MENU || -

Seleccione alguno de los siguientes métodos de ordenamiento:

[1]: Burbuja
[2]: Inserción directa
[3]: Selección directa
[4]: SALIR

///// 2_
```

```
a:\Principal\Escritorio\Probleuario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

///// MÉTODO DE ORDENAMIENTO: INSERCIÓN DIRECTA

Arreglo tamaño: 7

Ingrese los valores del arreglo:

[01] → -2
[02] → 23
[03] → 0
[04] → -12
[05] → 54
[06] → 9
[07] → 1

///// ARREGLO ORDENADO:

[01] → -12
[02] → -2
[03] → 0
[04] → 1
[05] → 9
[06] → 23
[07] → 54

¿Desea continuar? (S/N) >> s_
```

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

```
a:\Principal\Escritorio\Problematario 7\problema 1V2.exe
- - - ALGORITMOS DE ORDENAMIENTO - - -

- || MENU || -

Seleccione alguno de los siguientes métodos de ordenamiento:

[1]: Burbuja
[2]: Inserción directa
[3]: Selección directa
[4]: SALIR

///// 3_

a:\Principal\Escritorio\Problematario 7\problema 1V2.exe
- - - ALGORITMOS DE ORDENAMIENTO - - -

///// MÉTODO DE ORDENAMIENTO: SELECCIÓN DIRECTA

Indique el tamaño del arreglo: 1

[!] ERROR: Valores incorrectos [!]

a:\Principal\Escritorio\Problematario 7\problema 1V2.exe
- - - ALGORITMOS DE ORDENAMIENTO - - -

///// MÉTODO DE ORDENAMIENTO: SELECCIÓN DIRECTA

Arreglo tamaño: 9

Ingrese los valores del arreglo:

[01] → -2
[02] → 0
[03] → -11
[04] → 12
[05] → -11
[06] → 23
[07] → 7
[08] → -2
[09] → 5

///// ARREGLO ORDENADO:

[01] → -11
[02] → -11
[03] → -2
[04] → -2
[05] → 0
[06] → 5
[07] → 7
[08] → 12
[09] → 23

¿Desea continuar? (S/N) >> s_
```



```
a:\Principal\Escritorio\Problemario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

- || MENU || -

Seleccione alguno de los siguientes métodos de ordenamiento:

[1]: Burbuja
[2]: Inserción directa
[3]: Selección directa
[4]: SALIR

///// 5

[!] ERROR: Valores incorrectos [!].
```

```
a:\Principal\Escritorio\Problemario 7\problema 1V2.exe

- - - ALGORITMOS DE ORDENAMIENTO - - -

- || MENU || -

Seleccione alguno de los siguientes métodos de ordenamiento:

[1]: Burbuja
[2]: Inserción directa
[3]: Selección directa
[4]: SALIR

///// 4.
```

```
a:\Principal\Escritorio\Problemario 7\problema 1V2.exe

GRACIAS POR USAR NUESTROS SERVICIOS

Equipo 8 - BUAP
```

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PRUEBA DE ESCRITORIO: Para realizar la prueba de escritorio utilice la siguiente serie de datos: 51, 21, 3, 39, 80, 36, 17.

LINEA					Descripción
3 - 5					directiva de preprocesador
7 - 16					Prototipos
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
19	*INT	INT	INT		Definir variables
20				CHAR	
21 - 35					Ciclo Do - while
23					Subproceso “Menu(&op)”
MENU [40 - 61]					
L	Memoria				Descripción
	*op		error		
41			INT		Definir variables
42 - 59					Ciclo Do - while
43			0		error = 0
44 - 51					Impresiones de indicaciones de menú
52	1				Indicación “Burbuja”
53 - 58					Estructura de selección simple para “op” (*op < 1    *op > 4) -> Falso
59					Comparador Ciclo Do – While (error == 1) -> Falso
60					Limpiar pantalla: system("cls");
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
25 - 33			1		Estructura de selección simple para “op” (op != 4)->Verdad
27					Subproceso “size(&tam,op)”
SIZE [79 - 94]					
L	Memoria				Descripción
	op	*tam	error		
80	1		INT		Definir variables
81 - 92					Ciclo Do - while
82					Subproceso “nom(op)”

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

NOM [63 - 77]					
L	Memoria			Descripción	
	op 1				
64 - 65				Impresiones indicaciones superiores	
66 - 76				Estructura Switch(op)	
67 - 69				Ya que op = 1 -> case 1 Impresión indicador de selección	
SIZE [79 - 94]					
L	Memoria			Descripción	
	op 1	*tam	error		
83			0	error = 0	
84				Impresión de indicación del tamaño del arreglo	
85		7		Entrada del tamaño del arreglo en “tam” por referencia	
86 - 91				Estructura de selección simple para “*tam”, (*tam <= 1) -> Falso	
92				Comparador Ciclo Do – While, (error == 1) -> Falso	
93				Impresión de salto de linea	
MAIN [18 - 38]					
L	Memoria				Descripción
	a DIN	tam 7	op 1	des	
28					Especificar la dimensión dinámica de “a”, (int *)calloc(tam,sizeof(int))
30					Subproceso “lect(a,tam,op)”
LECT [96 - 106]					
L	Memoria				Descripción
	arreg[] a[]	tam 7	op 1	i INT	
97					Definir variables
98					Limpiar pantalla: system("cls");
99					Subproceso “nom(op)”
NOM [63 - 77]					
L	Memoria			Descripción	
	op 1				
64 - 65				Impresiones indicaciones superiores	
66 - 76				Estructura Switch(op)	
67 - 69				Ya que op = 1 -> case 1 Impresión indicador de selección	
LECT [96 - 106]					

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

L	Memoria				Descripción
	arreg[]	tam	op	i	
100	a[]	7	1		Impresión informativa del tamaño del arreglo
101					Impresión solicitud de ingreso de valores al arreglo
102-105				0	Estructura de repetición “Para”, primera evaluación
103					Impresión de símbolos auxiliares
104	[0]=51				Entrada 0 para el arreglo “arreg”
*102				1	Segunda evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[1]=21				Entrada 1 para el arreglo “arreg”
*102				2	Tercera evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[2]=3				Entrada 2 para el arreglo “arreg”
*102				3	Cuarta evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[3]=39				Entrada 3 para el arreglo “arreg”
*102				4	Quinta evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[4]=80				Entrada 4 para el arreglo “arreg”
*102				5	Sexta evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[5]=36				Entrada 5 para el arreglo “arreg”
*102				6	Séptima evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[6]=17				Entrada 6 para el arreglo “arreg”
*102				7	Octava evaluación, i++ y se compara si es menor que “tam” -> Falso
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
32		7	1		Subproceso “casos(op,a,tam)”
CASOS [159 - 172]					
L	Memoria			Descripción	
	arreg[]	tam	op		
160	a[]	7	1	Impresión indicación de arreglo ordenado	
161-171				Estructura Switch(op)	

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

162-164					Ya que op = 1 -> case 1	
163					Subproceso "burbu(arreg,tam)"	
BURBU [115 - 127]						
L	Memoria					Descripción
	arreg[]	tam	i	j	aux	
116	a[]	7	INT	INT	INT	Definir variables
117			0			Estructura de repetición "Para", primera evaluación [i]
**118				0		Estructura de repetición "Para", primera evaluación [j]
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(51>21)->Verdad
120					51	aux = arreg[j] -> aux = 51
121	[0]=21					arreg[j] = arreg[j + 1] -> arreg[j] = 21
122	[1]=51					arreg[j + 1] = aux -> arreg[j + 1] = 51
**118				1		Segunda evaluación, j++ y se compara si j es menor que "tam-1" -> Verdad
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(51>3)->Verdad
120					51	aux = arreg[j] -> aux = 51
121	[1]=3					arreg[j] = arreg[j + 1] -> arreg[j] = 3
122	[2]=51					arreg[j + 1] = aux -> arreg[j + 1] = 51
**118				2		Tercera evaluación, j++ y se compara si j es menor que "tam-1" -> Verdad
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(51>39)->Verdad
120					51	aux = arreg[j] -> aux = 51
121	[2]=39					arreg[j] = arreg[j + 1] -> arreg[j] = 39
122	[3]=51					arreg[j + 1] = aux -> arreg[j + 1] = 51
**118				3		Cuarta evaluación, j++ y se compara si j es menor que "tam-1" -> Verdad
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(51>80)->Falso
**118				4		Quinta evaluación, j++ y se compara si j es menor que "tam-1" -> Verdad
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(80>36)->Verdad
120					80	aux = arreg[j] -> aux = 80
121	[4]=36					arreg[j] = arreg[j + 1] -> arreg[j] = 36
122	[5]=80					arreg[j + 1] = aux -> arreg[j + 1] = 80
**118				5		Sexta evaluación, j++ y se compara si j es menor que "tam-1" -> Verdad
119						Estructura de selección simple (arreg[j] > arreg[j + 1]) =(80>17)->Verdad
120					80	aux = arreg[j] -> aux = 80
121	[5]=17					arreg[j] = arreg[j + 1] -> arreg[j] = 17
122	[6]=80					arreg[j + 1] = aux -> arreg[j + 1] = 80
**118				6		Séptima evaluación, j++ y se compara si j es menor que "tam-1" -> Falso

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

<b>*117</b>			1			<i>Segunda evaluación, i++ y se compara si i es menor que "tam-1" -&gt; Verdad</i>
<b>**118</b>				0		<i>Primera evaluación y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(21&gt;3)-&gt;Verdad</i>
<b>120</b>					21	<i>aux = arreg[j] -&gt; aux = 21</i>
<b>121</b>	[0]=3					<i>arreg[j] = arreg[j + 1] -&gt; arreg[j] = 3</i>
<b>122</b>	[1]=21					<i>arreg[j + 1] = aux -&gt; arreg[j + 1] = 21</i>
<b>**118</b>				1		<i>Segunda evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(21&gt;39)-&gt;Falso</i>
<b>**118</b>				2		<i>Tercera evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(39&gt;51)-&gt;Falso</i>
<b>**118</b>				3		<i>Cuarta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(51&gt;36)-&gt;Verdad</i>
<b>120</b>					51	<i>aux = arreg[j] -&gt; aux = 51</i>
<b>121</b>	[3]=36					<i>arreg[j] = arreg[j + 1] -&gt; arreg[j] = 36</i>
<b>122</b>	[4]=51					<i>arreg[j + 1] = aux -&gt; arreg[j + 1] = 51</i>
<b>**118</b>				4		<i>Quinta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(51&gt;17)-&gt;Verdad</i>
<b>120</b>					51	<i>aux = arreg[j] -&gt; aux = 51</i>
<b>121</b>	[4]=17					<i>arreg[j] = arreg[j + 1] -&gt; arreg[j] = 17</i>
<b>122</b>	[5]=51					<i>arreg[j + 1] = aux -&gt; arreg[j + 1] = 51</i>
<b>**118</b>				5		<i>Sexta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(51&gt;80)-&gt;Falso</i>
<b>**118</b>				6		<i>Séptima evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Falso</i>
<b>*117</b>			2			<i>Tercera evaluación, i++ y se compara si i es menor que "tam-1" -&gt; Verdad</i>
<b>**118</b>				0		<i>Primera evaluación y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(3&gt;21)-&gt;Falso</i>
<b>**118</b>				1		<i>Segunda evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(21&gt;39)-&gt;Falso</i>
<b>**118</b>				2		<i>Tercera evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(39&gt;36)-&gt;Verdad</i>
<b>120</b>					39	<i>aux = arreg[j] -&gt; aux = 39</i>
<b>121</b>	[2]=36					<i>arreg[j] = arreg[j + 1] -&gt; arreg[j] = 36</i>
<b>122</b>	[3]=39					<i>arreg[j + 1] = aux -&gt; arreg[j + 1] = 39</i>
<b>**118</b>				3		<i>Cuarta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(39&gt;17)-&gt;Verdad</i>

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

120				39	$aux = arreg[j] \rightarrow aux = 39$
121	[3]=17				$arreg[j] = arreg[j + 1] \rightarrow arreg[j] = 17$
122	[4]=39				$arreg[j + 1] = aux \rightarrow arreg[j + 1] = 39$
**118			4		Quinta evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (39>51)->Falso
**118			5		Sexta evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (51>80)->Falso
**118			6		Séptima evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Falso
*117		3			Cuarta evaluación, $i++$ y se compara si $i$ es menor que "tam-1" -> Verdad
**118			0		Primera evaluación y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (3>21)->Falso
**118			1		Segunda evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (21>36)->Falso
**118			2		Tercera evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (36>17)->Verdad
120				36	$aux = arreg[j] \rightarrow aux = 36$
121	[2]=17				$arreg[j] = arreg[j + 1] \rightarrow arreg[j] = 17$
122	[3]=36				$arreg[j + 1] = aux \rightarrow arreg[j + 1] = 36$
**118			3		Cuarta evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (36>39)->Falso
**118			4		Quinta evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (39>51)->Falso
**118			5		Sexta evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (51>80)->Falso
**118			6		Séptima evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Falso
*117		4			Quinta evaluación, $i++$ y se compara si $i$ es menor que "tam-1" -> Verdad
**118			0		Primera evaluación y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (3>21)->Falso
**118			1		Segunda evaluación, $j++$ y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (21>17)->Verdad
120				21	$aux = arreg[j] \rightarrow aux = 21$
121	[1]=17				$arreg[j] = arreg[j + 1] \rightarrow arreg[j] = 17$
122	[2]=21				$arreg[j + 1] = aux \rightarrow arreg[j + 1] = 21$
**118			2		Tercera evaluación y se compara si $j$ es menor que "tam-1" -> Verdad
119					Estructura de selección simple ( $arreg[j] > arreg[j + 1]$ ) = (21>36)->Falso



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

<b>**118</b>				3		<i>Cuarta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(36&gt;39)-&gt;Falso</i>
<b>**118</b>				4		<i>Quinta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(39&gt;51)-&gt;Falso</i>
<b>**118</b>				5		<i>Sexta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(51&gt;80)-&gt;Falso</i>
<b>**118</b>				6		<i>Séptima evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Falso</i>
<b>*117</b>			5			<i>Sexta evaluación, i++ y se compara si i es menor que "tam-1" -&gt; Verdad</i>
<b>**118</b>				0		<i>Primera evaluación y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(3&gt;17)-&gt;Falso</i>
<b>**118</b>				1		<i>Segunda evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(17&gt;21)-&gt;Falso</i>
<b>**118</b>				2		<i>Tercera evaluación y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(21&gt;36)-&gt;Falso</i>
<b>**118</b>				3		<i>Cuarta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(36&gt;39)-&gt;Falso</i>
<b>**118</b>				4		<i>Quinta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(39&gt;51)-&gt;Falso</i>
<b>**118</b>				5		<i>Sexta evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Verdad</i>
<b>119</b>						<i>Estructura de selección simple (arreg[j] &gt; arreg[j + 1]) =(51&gt;80)-&gt;Falso</i>
<b>**118</b>				6		<i>Séptima evaluación, j++ y se compara si j es menor que "tam-1" -&gt; Falso</i>
<b>*117</b>			6			<i>Séptima evaluación, i++ y se compara si i es menor que "tam-1" -&gt; Falso</i>
<b>126</b>						<i>Subproceso "impr(arreg,tam)"</i>
<b>IMPR [108 - 113]</b>						
L	Memoria			Descripción		
	arreg[]	tam	i			
<b>109</b>	a[]	7	INT	<i>Definir variables</i>		
<b>*110</b>			0	<i>Estructura de repetición "Para", primera evaluación</i>		
<b>111</b>				<i>Impresión de indicaciones auxiliares y el valor del arreglo en [0]</i>		
<b>*110</b>			1	<i>Segunda evaluación, i++ y se compara si i es menor que "tam" -&gt; Verdad</i>		
<b>111</b>				<i>Impresión de indicaciones auxiliares y el valor del arreglo en [1]</i>		
<b>*110</b>			2	<i>Tercera evaluación, i++ y se compara si i es menor que "tam" -&gt; Verdad</i>		
<b>111</b>				<i>Impresión de indicaciones auxiliares y el valor del arreglo en [2]</i>		
<b>*110</b>			3	<i>Cuarta evaluación, i++ y se compara si i es menor que "tam" -&gt; Verdad</i>		
<b>111</b>				<i>Impresión de indicaciones auxiliares y el valor del arreglo en [3]</i>		



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

*110			4	Quinta evaluación, i++ y se compara si i es menor que “tam” -> Verdad	
111				Impresión de indicaciones auxiliares y el valor del arreglo en [4]	
*110			5	Sexta evaluación, i++ y se compara si i es menor que “tam” -> Verdad	
111				Impresión de indicaciones auxiliares y el valor del arreglo en [5]	
*110			6	Séptima evaluación, i++ y se compara si i es menor que “tam” -> Verdad	
111				Impresión de indicaciones auxiliares y el valor del arreglo en [6]	
*110			7	Octava evaluación, i++ y se compara si i es menor que “tam” -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
34		7	1		Subproceso “salida(&des,op)”
SALIDA [174 - 200]					
L	Memoria			Descripción	
	*des	op	error		
175		1	INT	Definir variables	
176-199				Ciclo Do - while	
177			0	Error = 0	
178-184				estructura de selección doble (op == 4) -> Falso	
181-184				else	
182				Impresión indicación sobre continuar en el programa	
183	s			Entrada de “des” por referencia	
185-198				estructura de selección doble (*des == 's'    *des == 'S') -> Verdad	
186				System(“cls”)	
199				Comparador Ciclo Do – While, (error == 1) -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
35		7	1	s	Comparador Ciclo Do – While, (des == 's'    des == 'S') -> Verdad (bucle)
21 - 35					Ciclo Do - while
23					Subproceso “Menu(&op)”
MENU [40 - 61]					
L	Memoria			Descripción	
	*op	error			
41	7	INT		Definir variables	
42 - 59				Ciclo Do - while	

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

43		0		<i>error = 0</i>	
44 - 51				<i>Impresiones de indicaciones de menú</i>	
52	2			<i>Indicación “Inserción directa”</i>	
53 - 58				<i>Estructura de selección simple para “op” (*op &lt; 1    *op &gt; 4) -&gt; Falso</i>	
59				<i>Comparador Ciclo Do – While (error == 1) -&gt; Falso</i>	
60				<i>Limpiar pantalla: system("cls");</i>	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
25 - 33		7	2	s	<i>Estructura de selección simple para “op” (op != 4)-&gt;Verdad</i>
27					<i>Subproceso “size(&amp;tam,op)”</i>
SIZE [79 - 94]					
L	Memoria			Descripción	
	op	*tam	error		
80	2	7	INT	<i>Definir variables</i>	
81 - 92				<i>Ciclo Do - while</i>	
82				<i>Subproceso “nom(op)”</i>	
NOM [63 - 77]					
L	Memoria			Descripción	
	op				
64 - 65	2			<i>Impresiones indicaciones superiores</i>	
66 - 76				<i>Estructura Switch(op)</i>	
67 - 69				<i>Ya que op = 2 -&gt; case 2</i> <i>Impresión indicador de selección</i>	
SIZE [79 - 94]					
L	Memoria			Descripción	
	op	*tam	error		
83	2	7	0	<i>error = 0</i>	
84				<i>Impresión de indicación del tamaño del arreglo</i>	
85		7		<i>Entrada del tamaño del arreglo en “tam” por referencia</i>	
86 - 91				<i>Estructura de selección simple para “*tam”, (*tam &lt;= 1) -&gt; Falso</i>	
92				<i>Comparador Ciclo Do – While, (error == 1) -&gt; Falso</i>	
93				<i>Impresión de salto de linea</i>	
MAIN [18 - 38]					
L	Memoria			Descripción	

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

	a	tam	op	des	
28	DIN	7	2	S	Especificar la dimensión dinámica de “a”, (int *)calloc(tam,sizeof(int))
30					Subproceso “lect(a,tam,op)”
LECT [96 - 106]					
L	Memoria				Descripción
	arreg[]	tam	op	i	
97	a[]	7	2	INT	Definir variables
98					Limpiar pantalla: system("cls");
99					Subproceso “nom(op)”
NOM [63 - 77]					
L	Memoria				Descripción
	op				
64 - 65	2				Impresiones indicaciones superiores
66 - 76					Estructura Switch(op)
67 - 69					Ya que op = 2 -> case 2 Impresión indicador de selección
LECT [96 - 106]					
L	Memoria				Descripción
	arreg[]	tam	op	i	
100	a[]	7	2		Impresión informativa del tamaño del arreglo
101					Impresión solicitud de ingreso de valores al arreglo
102-105				0	Estructura de repetición “Para”, primera evaluación
103					Impresión de símbolos auxiliares
104	[0]=51				Entrada 0 para el arreglo “arreg”
*102				1	Segunda evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[1]=21				Entrada 1 para el arreglo “arreg”
*102				2	Tercera evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[2]=3				Entrada 2 para el arreglo “arreg”
*102				3	Cuarta evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares
104	[3]=39				Entrada 3 para el arreglo “arreg”
*102				4	Quinta evaluación, i++ y se compara si es menor que “tam” -> Verdad
103					Impresión de símbolos auxiliares

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

104	[4]=80				Entrada 4 para el arreglo "arreg"	
*102				5	Sexta evaluación, i++ y se compara si es menor que "tam" -> Verdad	
103					Impresión de símbolos auxiliares	
104	[5]=36				Entrada 5 para el arreglo "arreg"	
*102				6	Séptima evaluación, i++ y se compara si es menor que "tam" -> Verdad	
103					Impresión de símbolos auxiliares	
104	[6]=17				Entrada 6 para el arreglo "arreg"	
*102				7	Octava evaluación, i++ y se compara si es menor que "tam" -> Falso	
MAIN [18 - 38]						
L	Memoria				Descripción	
	a	tam	op	des		
32		7	2	s	Subproceso "casos(op,a,tam)"	
CASOS [159 - 172]						
L	Memoria			Descripción		
	arreg[]	tam	op			
160	a[]	7	2	Impresión indicación de arreglo ordenado		
161-171				Estructura Switch(op)		
162-164				Ya que op = 2 -> case 2		
163				Subproceso "indi(arreg,tam)"		
INDI [129 - 141]						
L	Memoria					Descripción
	arreg[]	tam	i	act	aux	
130	a[]	7	INT	INT	INT	Definir variables
*131			0			Estructura de repetición "Para", primera evaluación
132				0		act = i
**133						Estructura While, (act > 0 && arreg[act] < arreg[act - 1]) -> Falso
*131			1			Segunda evaluación, i++ y se compara si es menor que "tam" -> Verdad
132				1		act = i
**133						Estructura While, (act > 0 && arreg[act] < arreg[act - 1]) -> (1 > 0 && 21 < 51) -> Verdad
134					51	aux = arreg[act - 1] -> aux = 51
135	[0]=21					arreg[act - 1] = arreg[act] -> arreg[act - 1] = 21
136	[1]=51					arreg[act] = aux -> arreg[act] = 51
137				0		act = act - 1

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (0 &gt; 0 &amp;&amp; 51 &lt; ??) -&gt; Falso</i>
*131			2			<i>Tercera evaluación, i++ y se compara si es menor que "tam" -&gt; Verdad</i>
132				2		<i>act = i</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (2 &gt; 0 &amp;&amp; 3 &lt; 51) -&gt; Verdad</i>
134					51	<i>aux = arreg[act - 1] -&gt; aux = 51</i>
135	[1]=3					<i>arreg[act - 1] = arreg[act] -&gt; arreg[act - 1] = 3</i>
136	[2]=51					<i>arreg[act] = aux -&gt; arreg[act] = 51</i>
137				1		<i>act = act - 1</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (1 &gt; 0 &amp;&amp; 3 &lt; 21) -&gt; Verdad</i>
134					21	<i>aux = arreg[act - 1] -&gt; aux = 21</i>
135	[0]=3					<i>arreg[act - 1] = arreg[act] -&gt; arreg[act - 1] = 3</i>
136	[1]=21					<i>arreg[act] = aux -&gt; arreg[act] = 21</i>
137				0		<i>act = act - 1</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (0 &gt; 0 &amp;&amp; 3 &lt; ??) -&gt; Falso</i>
*131			3			<i>Cuarta evaluación, i++ y se compara si es menor que "tam" -&gt; Verdad</i>
132				3		<i>act = i</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (3 &gt; 0 &amp;&amp; 39 &lt; 51) -&gt; Verdad</i>
134					51	<i>aux = arreg[act - 1] -&gt; aux = 51</i>
135	[2]=39					<i>arreg[act - 1] = arreg[act] -&gt; arreg[act - 1] = 39</i>
136	[3]=51					<i>arreg[act] = aux -&gt; arreg[act] = 51</i>
137				2		<i>act = act - 1</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (2 &gt; 0 &amp;&amp; 39 &lt; 21) -&gt; Falso</i>
*131			4			<i>Quinta evaluación, i++ y se compara si es menor que "tam" -&gt; Verdad</i>
132				4		<i>act = i</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (4 &gt; 0 &amp;&amp; 80 &lt; 51) -&gt; Falso</i>
*131			5			<i>Sexta evaluación, i++ y se compara si es menor que "tam" -&gt; Verdad</i>
132				5		<i>act = i</i>
**133						<i>Estructura While, (act &gt; 0 &amp;&amp; arreg[act] &lt; arreg[act - 1]) -&gt; (5 &gt; 0 &amp;&amp; 36 &lt; 80) -&gt; Verdad</i>

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

134				80	$aux = arreg[act - 1] \rightarrow aux = 80$
135	[4]=36				$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 36$
136	[5]=80				$arreg[act] = aux \rightarrow arreg[act] = 80$
137			4		$act = act - 1$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (4 &gt; 0 \ \&amp;\&amp; \ 36 &lt; 51) \rightarrow Verdad</math></i>
134				51	$aux = arreg[act - 1] \rightarrow aux = 51$
135	[3]=36				$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 36$
136	[4]=51				$arreg[act] = aux \rightarrow arreg[act] = 51$
137			3		$act = act - 1$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (3 &gt; 0 \ \&amp;\&amp; \ 36 &lt; 39) \rightarrow Verdad</math></i>
134				39	$aux = arreg[act - 1] \rightarrow aux = 39$
135	[2]=36				$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 36$
136	[3]=39				$arreg[act] = aux \rightarrow arreg[act] = 39$
137			2		$act = act - 1$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (2 &gt; 0 \ \&amp;\&amp; \ 36 &lt; 21) \rightarrow Falso</math></i>
*131		6			<i>séptima evaluación, <math>i++</math> y se compara si es menor que "tam" <math>\rightarrow Verdad</math></i>
132			6		$act = i$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (6 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 80) \rightarrow Verdad</math></i>
134				80	$aux = arreg[act - 1] \rightarrow aux = 80$
135	[5]=17				$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 17$
136	[6]=80				$arreg[act] = aux \rightarrow arreg[act] = 80$
137			5		$act = act - 1$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (5 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 51) \rightarrow Verdad</math></i>
134				51	$aux = arreg[act - 1] \rightarrow aux = 51$
135	[4]=17				$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 17$
136	[5]=51				$arreg[act] = aux \rightarrow arreg[act] = 51$
137			4		$act = act - 1$
**133					<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (4 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 39) \rightarrow Verdad</math></i>
134				39	$aux = arreg[act - 1] \rightarrow aux = 39$

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

135	[3]=17					$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 17$
136	[4]=39					$arreg[act] = aux \rightarrow arreg[act] = 39$
137				3		$act = act - 1$
**133						<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (3 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 36) \rightarrow Verdad</math></i>
134					36	$aux = arreg[act - 1] \rightarrow aux = 36$
135	[2]=17					$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 17$
136	[3]=36					$arreg[act] = aux \rightarrow arreg[act] = 36$
137				2		$act = act - 1$
**133						<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (2 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 21) \rightarrow Verdad</math></i>
134					21	$aux = arreg[act - 1] \rightarrow aux = 21$
135	[1]=17					$arreg[act - 1] = arreg[act] \rightarrow arreg[act - 1] = 17$
136	[2]=21					$arreg[act] = aux \rightarrow arreg[act] = 21$
137				1		$act = act - 1$
**133						<i>Estructura While, <math>(act &gt; 0 \ \&amp;\&amp; \ arreg[act] &lt; arreg[act - 1]) \rightarrow (2 &gt; 0 \ \&amp;\&amp; \ 17 &lt; 3) \rightarrow Falso</math></i>
*131			7			<i>Octava evaluación, <math>i++</math> y se compara si es menor que "tam" <math>\rightarrow Falso</math></i>
140						<i>Subproceso "impr(arreg,tam)"</i>
IMPR [108 - 113]						
L	Memoria			Descripción		
	arreg[]	tam	i			
109	a[]	7	INT	Definir variables		
*110			0	Estructura de repetición "Para", primera evaluación		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [0]		
*110			1	Segunda evaluación, $i++$ y se compara si $i$ es menor que "tam" $\rightarrow Verdad$		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [1]		
*110			2	Tercera evaluación, $i++$ y se compara si $i$ es menor que "tam" $\rightarrow Verdad$		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [2]		
*110			3	Cuarta evaluación, $i++$ y se compara si $i$ es menor que "tam" $\rightarrow Verdad$		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [3]		
*110			4	Quinta evaluación, $i++$ y se compara si $i$ es menor que "tam" $\rightarrow Verdad$		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [4]		
*110			5	Sexta evaluación, $i++$ y se compara si $i$ es menor que "tam" $\rightarrow Verdad$		
111				Impresión de indicaciones auxiliares y el valor del arreglo en [5]		

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

*110			6	Séptima evaluación, i++ y se compara si i es menor que “tam” -> Verdad	
111				Impresión de indicaciones auxiliares y el valor del arreglo en [6]	
*110			7	Octava evaluación, i++ y se compara si i es menor que “tam” -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
34		7	2		Subproceso “salida(&des,op)”
SALIDA [174 - 200]					
L	Memoria			Descripción	
	*des	op	error		
175		2	INT	Definir variables	
176-199				Ciclo Do - while	
177			0	Error = 0	
178-184				estructura de selección doble (op == 4) -> Falso	
181-184				else	
182				Impresión indicación sobre continuar en el programa	
183	s			Entrada de “des” por referencia	
185-198				estructura de selección doble (*des == 's'    *des == 'S') -> Verdad	
186				System(“cls”)	
199				Comparador Ciclo Do – While, (error == 1) -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
35		7	2	s	Comparador Ciclo Do – While, (des == 's'    des == 'S') -> Verdad (bucle)
21 - 35					Ciclo Do - while
23					Subproceso “Menu(&op)”
MENU [40 - 61]					
L	Memoria			Descripción	
	*op	error			
41	7	INT		Definir variables	
42 - 59				Ciclo Do - while	
43		0		error = 0	
44 - 51				Impresiones de indicaciones de menú	
52	3			Indicación “Selección directa”	
53 - 58				Estructura de selección simple para “op” (*op < 1    *op > 4) -> Falso	



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

59					Comparador Ciclo Do – While (error == 1) -> Falso
60					Limpiar pantalla: system("cls");
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
25 - 33		7	3	s	Estructura de selección simple para “op” (op != 4)->Verdad
27					Subproceso “size(&tam,op)”
SIZE [79 - 94]					
L	Memoria			Descripción	
	op	*tam	error		
80	3	7	INT	Definir variables	
81 - 92				Ciclo Do - while	
82				Subproceso “nom(op)”	
NOM [63 - 77]					
L	Memoria			Descripción	
	op				
64 - 65	3			Impresiones indicaciones superiores	
66 - 76				Estructura Switch(op)	
67 - 69				Ya que op = 3 -> case 3 Impresión indicador de selección	
SIZE [79 - 94]					
L	Memoria			Descripción	
	op	*tam	error		
83	3	7	0	error = 0	
84				Impresión de indicación del tamaño del arreglo	
85		7		Entrada del tamaño del arreglo en “tam” por referencia	
86 - 91				Estructura de selección simple para “*tam”, (*tam <= 1) -> Falso	
92				Comparador Ciclo Do – While, (error == 1) -> Falso	
93				Impresión de salto de linea	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
28	DIN	7	3	s	Especificar la dimensión dinámica de “a”, (int *)calloc(tam,sizeof(int))
30					Subproceso “lect(a,tam,op)”
LECT [96 - 106]					

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

L	Memoria				Descripción
	arreg[]	tam	op	i	
97	a[]	7	3	INT	Definir variables
98					Limpiar pantalla: system("cls");
99					Subproceso "nom(op)"
NOM [63 - 77]					
L	Memoria				Descripción
	op				
64 - 65	3				Impresiones indicaciones superiores
66 - 76					Estructura Switch(op)
67 - 69					Ya que op = 3 -> case 3 Impresión indicador de selección
LECT [96 - 106]					
L	Memoria				Descripción
	arreg[]	tam	op	i	
100	a[]	7	3		Impresión informativa del tamaño del arreglo
101					Impresión solicitud de ingreso de valores al arreglo
102-105				0	Estructura de repetición "Para", primera evaluación
103					Impresión de símbolos auxiliares
104	[0]=51				Entrada 0 para el arreglo "arreg"
*102				1	Segunda evaluación, i++ y se compara si es menor que "tam" -> Verdad
103					Impresión de símbolos auxiliares
104	[1]=21				Entrada 1 para el arreglo "arreg"
*102				2	Tercera evaluación, i++ y se compara si es menor que "tam" -> Verdad
103					Impresión de símbolos auxiliares
104	[2]=3				Entrada 2 para el arreglo "arreg"
*102				3	Cuarta evaluación, i++ y se compara si es menor que "tam" -> Verdad
103					Impresión de símbolos auxiliares
104	[3]=39				Entrada 3 para el arreglo "arreg"
*102				4	Quinta evaluación, i++ y se compara si es menor que "tam" -> Verdad
103					Impresión de símbolos auxiliares
104	[4]=80				Entrada 4 para el arreglo "arreg"
*102				5	Sexta evaluación, i++ y se compara si es menor que "tam" -> Verdad
103					Impresión de símbolos auxiliares
104	[5]=36				Entrada 5 para el arreglo "arreg"

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

*102				6	Séptima evaluación, i++ y se compara si es menor que “tam” -> Verdad		
103					Impresión de símbolos auxiliares		
104	[6]=17				Entrada 6 para el arreglo “arreg”		
*102				7	Octava evaluación, i++ y se compara si es menor que “tam” -> Falso		
MAIN [18 - 38]							
L	Memoria				Descripción		
	a	tam	op	des			
32		7	3	s	Subproceso “casos(op,a,tam)”		
CASOS [159 - 172]							
L	Memoria			Descripción			
	arreg[]	tam	op				
160	a[]	7	3	Impresión indicación de arreglo ordenado			
161-171				Estructura Switch(op)			
162-164				Ya que op = 3 -> case 3			
163				Subproceso “direct(arreg,tam)”			
DIRECT [143 - 157]							
L	Memoria						Descripción
	arreg[]	tam	i	j	min	aux	
144	a[]	7	INT	INT	INT	INT	Definir variables
*145			0				Estructura de repetición “Para”, primera evaluación [i]
146					0		min = i
**147				1			Estructura de repetición “Para”, primera evaluación [j]
148							Estructura de selección simple(arreg[j] < arreg[min]) -> (21<51) -> Verdad
149					1		min = j
**147				2			Segunda evaluación, j++ y se compara si es menor que “tam” -> Verdad
148							Estructura de selección simple(arreg[j] < arreg[min]) -> (3<21) -> Verdad
149					2		min = j
**147				3			Tercera evaluación, j++ y se compara si es menor que “tam” -> Verdad
148							Estructura de selección simple(arreg[j] < arreg[min]) -> (39<3) -> Falso

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**147				4			Cuarta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(80 < 3)$ -> Falso
**147				5			Quinta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(36 < 3)$ -> Falso
**147				6			Sexta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(17 < 3)$ -> Falso
**147				7			Séptima evaluación, $j++$ y se compara si es menor que "tam" -> Falso
152						51	$\text{aux} = \text{arreg}[i]$
153	[0]=3						$\text{arreg}[i] = \text{arreg}[\text{min}]$
154	[2]=51						$\text{arreg}[\text{min}] = \text{aux}$
*145			1				Segunda evaluación, $i++$ y se compara si es menor que "tam-1" -> Verdad
146					1		$\text{min} = i$
**147				2			Estructura de repetición "Para", primera evaluación [j]
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(51 < 21)$ -> Falso
**147				3			Segunda evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(39 < 21)$ -> Falso
**147				4			Tercera evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(80 < 21)$ -> Falso
**147				5			Cuarta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> $(36 < 21)$ -> Falso

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**147				6			Quinta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> (17<21) -> Verdad
149					6		$\text{min} = j$
**147				7			Sexta evaluación, $j++$ y se compara si es menor que "tam" -> Falso
152						21	$\text{aux} = \text{arreg}[i]$
153	[1]=17						$\text{arreg}[i] = \text{arreg}[\text{min}]$
154	[6]=21						$\text{arreg}[\text{min}] = \text{aux}$
*145			2				Tercera evaluación, $i++$ y se compara si es menor que "tam-1" -> Verdad
146					2		$\text{min} = i$
**147				3			Estructura de repetición "Para", primera evaluación [j]
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> (39<51) -> Verdad
149					3		$\text{min} = j$
**147				4			Segunda evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> (80<39) -> Falso
**147				5			Tercera evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> (36<39) -> Verdad
149					5		$\text{min} = j$
**147				6			Cuarta evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $\text{arreg}[j] < \text{arreg}[\text{min}]$ ) -> (21<36) -> Verdad
149					6		$\text{min} = j$
**147				7			Quinta evaluación, $j++$ y se compara si es menor que "tam" -> Falso
152						51	$\text{aux} = \text{arreg}[i]$
153	[2]=21						$\text{arreg}[i] = \text{arreg}[\text{min}]$
154	[6]=51						$\text{arreg}[\text{min}] = \text{aux}$

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

*145			3				Cuarta evaluación, $i++$ y se compara si es menor que "tam-1" -> Verdad
146					3		$min = i$
**147				4			Estructura de repetición "Para", primera evaluación [j]
148							Estructura de selección simple( $arreg[j] < arreg[min]$ ) -> (80<39) -> Falso
**147				5			Segunda evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $arreg[j] < arreg[min]$ ) -> (39<39) -> Verdad
149					5		$min = j$
**147				6			Tercera evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $arreg[j] < arreg[min]$ ) -> (51<36) -> Falso
**147				7			Cuarta evaluación, $j++$ y se compara si es menor que "tam" -> Falso
152						39	$aux = arreg[i]$
153	[3]=36						$arreg[i] = arreg[min]$
154	[5]=39						$arreg[min] = aux$
*145			4				Quinta evaluación, $i++$ y se compara si es menor que "tam-1" -> Verdad
146					4		$min = i$
**147				5			Estructura de repetición "Para", primera evaluación [j]
148							Estructura de selección simple( $arreg[j] < arreg[min]$ ) -> (39<80) -> Verdad
149					5		$min = j$
**147				6			Segunda evaluación, $j++$ y se compara si es menor que "tam" -> Verdad
148							Estructura de selección simple( $arreg[j] < arreg[min]$ ) -> (51<39) -> Falso
**147				7			Tercera evaluación, $j++$ y se compara si es menor que "tam" -> Falso
152						80	$aux = arreg[i]$
153	[4]=39						$arreg[i] = arreg[min]$
154	[5]=80						$arreg[min] = aux$

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

*145			5				Sexta evaluación, i++ y se compara si es menor que “tam-1” -> Verdad
146					5		min = i
**147				6			Estructura de repetición “Para”, primera evaluación [j]
148							Estructura de selecció simple(arreg[j] < arreg[min]) -> (51<80) -> Verdad
149					6		min = j
**147				7			Segunda evaluación, j++ y se compara si es menor que “tam” -> Falso
152						80	aux = arreg[i]
153	[5]=51						arreg[i] = arreg[min]
154	[6]=80						arreg[min] = aux
*145			6				Séptima evaluación, i++ y se compara si es menor que “tam-1” -> Falso
156							Subproceso “impr(arreg,tam)”
IMPR [108 - 113]							
L	Memoria			Descripción			
	arreg[]	tam	i				
109	a[]	7	INT	Definir variables			
*110			0	Estructura de repetición “Para”, primera evaluación			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [0]			
*110			1	Segunda evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [1]			
*110			2	Tercera evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [2]			
*110			3	Cuarta evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [3]			
*110			4	Quinta evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [4]			
*110			5	Sexta evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [5]			
*110			6	Séptima evaluación, i++ y se compara si i es menor que “tam” -> Verdad			
111				Impresión de indicaciones auxiliares y el valor del arreglo en [6]			
*110			7	Octava evaluación, i++ y se compara si i es menor que “tam” -> Falso			
MAIN [18 - 38]							

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

L	Memoria				Descripción
	a	tam	op	des	
34		7	3		Subproceso “salida(&des,op)”
SALIDA [174 - 200]					
L	Memoria			Descripción	
	*des	op	error		
175		3	INT	Definir variables	
176-199				Ciclo Do - while	
177			0	Error = 0	
178-184				estructura de selección doble (op == 4) -> Falso	
181-184				else	
182				Impresión indicación sobre continuar en el programa	
183	s			Entrada de “des” por referencia	
185-198				estructura de selección doble (*des == 's'    *des == 'S') -> Verdad	
186				System(“cls”)	
199				Comparador Ciclo Do – While, (error == 1) -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
35		7	3	s	Comparador Ciclo Do – While, (des == 's'    des == 'S') -> Verdad (bucle)
21 - 35					Ciclo Do - while
23					Subproceso “Menu(&op)”
MENU [40 - 61]					
L	Memoria		error	Descripción	
	*op				
41	7		INT	Definir variables	
42 - 59				Ciclo Do - while	
43			0	error = 0	
44 - 51				Impresiones de indicaciones de menú	
52	4			Indicación “SALIR”	
53 - 58				Estructura de selección simple para “op” (*op < 1    *op > 4) -> Falso	
59				Comparador Ciclo Do – While (error == 1) -> Falso	
60				Limpiar pantalla: system(“cls”);	
MAIN [18 - 38]					
L	Memoria			Descripción	



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

	a	tam	op	des	
25 - 33		7	4	s	Estructura de selección simple para “op” (op != 4)->Falso
27					Subproceso “salida(&des,op)”
SALIDA [174 - 200]					
L	Memoria			Descripción	
	*des	op	error		
175		4	INT	Definir variables	
176-199				Ciclo Do - while	
177			0	Error = 0	
178-184				estructura de selección doble (op == 4) -> Verdad	
179	N			*des = ‘N’	
185-198				estructura de selección doble (*des == 's'    *des == 'S') -> Falso	
188				Else	
189-197				estructura de selección doble (*des == 'n'    *des == 'N') -> Verdadero	
190-192				Impresión de despedida y credits	
199				Comparador Ciclo Do – While (error == 1) -> Falso	
MAIN [18 - 38]					
L	Memoria				Descripción
	a	tam	op	des	
35		7	4	N	Comparador Ciclo Do – While (des == 's'    des == 'S') -> Falso
36					getch()
37					return 0

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

2. Implemente un programa en lenguaje C que programe los algoritmos de búsqueda:
  - a. Lineal
  - b. Binaria

Mediante un menú se debe seleccionar el algoritmo para buscar un dato.

Al seleccionar un algoritmo se pedirá el tamaño del arreglo, el arreglo y el dato a buscar.

Se mostrará el resultado con la posición del dato, si se encontró o bien el mensaje de dato no encontrado.

Para la búsqueda binaria se debe ordenar el arreglo.

Solo saldrá del programa con la opción de salida.

CODIGO:

```
1. /*2.Algoritmos de busqueda lineal y binaria*/
2.
3. //cargar librerias
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <conio.h>
7.
8. //Prototipos
9. void menu(int *);
10. void nombre(int);
11. void dimension(int *,int);
12. void valores (int [],int,int);
13. void impresion(int [],int);
14. void metodo(int,int [],int,int);
15. void soli (int*);
16. void lineal(int,int [],int);
17. void binaria(int [],int,int);
18. void final(char *,int);
19.
20. //Principal
21. int main (){
22.     int x,op,longi,*arreglo;
23.     char op2;
24.     do{
25.         menu(&op);
26.         if (op!=3){
27.             dimension(&longi,op);
28.             arreglo=(int *)calloc(longi,sizeof(int));
29.             valores(arreglo,longi,op);
30.             soli(&x);
31.             metodo(op,arreglo,longi,x);
32.         }
```



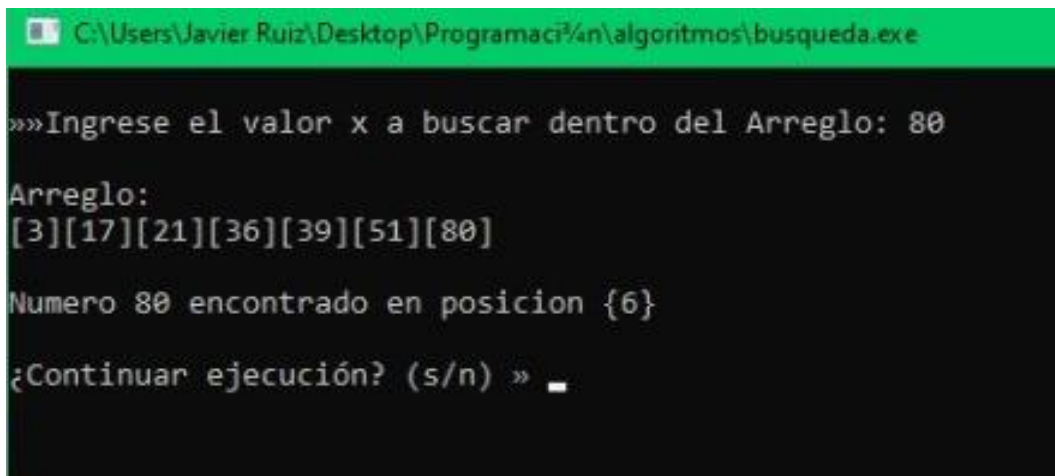
```
78. //funcion de tipo void para dimensionar el arreglo
79. void dimension (int *longi,int op){
80.     int error;
81.     do{
82.         nombre(op);
83.         system ("cls");
84.         error = 0;
85.         printf("\n%c%cIndique el tama%co del Arreglo: ",175,175,164);
86.         scanf("%d",longi);
87.         if (*longi == 1 || *longi == 0){
88.             printf("\n[!] ERROR: Tama%co del Arreglo menor que 2 [!]",164);
89.             error=1;
90.             getch();
91.             system("cls");
92.         }else if (*longi < 0 ){
93.             printf ("\n[!] ERROR: Tama%co del Arreglo inexistente [!]",164);
94.             error=1;
95.             getch();
96.             system ("cls");
97.         }
98.     } while (error!=0);
99.     printf("\n");
100. }
101.
102. //funcion de tipo void para ingresar valores al arreglo
103. void valores(int A[],int longi,int op){
104.     int i;
105.     system("cls");
106.     nombre(op);
107.     printf("\n%cTama%co del Arreglo: %d\n\n",175,164,longi);
108.     printf("%c%cProceda a ingresar los valores en el Arreglo%c%c\n\n",175,175,174,174);
109.     for (i = 0; i < longi; i++){
110.         printf("\t%cValor %c%d en el Arreglo: ",219,35,i+1);
111.         scanf("%d",&A[i]);
112.     }
113. }
114.
115. //funcion de tipo void para imprimir los valores dentro del arreglo
116. void impresion(int A[],int longi){
117.     int i;
118.     printf ("Arreglo:\n");
119.     for (i = 0; i < longi; i++){
120.         printf("[%d]",A[i]);
121.     }
122. }
```

```
123.
124.     //funcion de tipo void para solicitar el valor a buscar dentro del arreg
    lo
125.     void soli (int *x){
126.         system ("cls");
127.         printf ("\n%c%cIngrese el valor x a buscar dentro del Arreglo: ",175
    ,175);
128.         scanf ("%d",x);
129.     }
130.
131.     //funcion de tipo void para metodo lineal
132.     void lineal(int longi,int A[],int x){
133.         int i,buscador;
134.         printf("\n");
135.         impresion(A,longi);
136.         for (i=0;i<=longi;i++){
137.             if(A[i]==x){
138.                 buscador=i;
139.                 break;
140.             }else{
141.                 buscador=-1;
142.             }
143.         }
144.         if (buscador== -1){
145.             printf("\n\nNumero no encontrado\n");
146.         }else{
147.             printf("\n\nNumero %d encontrado en la posicion {%d}\n",x,i);
148.         }
149.     }
150.     //funcion de tipo void para metodo binario
151.     void binaria(int A[],int longi,int x){
152.         int i,j,aux,mitad;
153.         int a=0,contadorA=0,contadorB=0;
154.         for (i = 0; i < longi; i++){
155.             for (j = i + 1; j < longi; j++){
156.                 if (A[j] < A[i]){
157.                     aux = A[i];
158.                     A[i] = A[j];
159.                     A[j] = aux;
160.                 }
161.             }
162.         }
163.         printf("\n");
164.         impresion(A,longi);
165.
166.         while (a <= longi){
```

```
167.         contadorA++;
168.         mitad = (a + longi) / 2;
169.         if(x > 200){
170.             printf("\n\nNumero no encontrado\n");
171.             break;
172.         }
173.         if(x == A[mitad]){
174.             printf("\n\nNumero %d encontrado en posicion {%d}\n", A[mitad], mitad);
175.             break;
176.         }else if(x < A[mitad]){
177.             longi = mitad -1;
178.         }else{
179.             a = mitad + 1;
180.         }
181.         contadorB++;
182.     }
183.     if(contadorA == contadorB){
184.         printf("\n\nNumero no encontrado\n");
185.     }
186. }
187.
188. //funcion de tipo void para seleccion de metodo
189. void metodo(int op,int A[],int longi,int x){
190.
191.     switch (op){
192.     case 1:
193.         lineal(longi,A,x);
194.         break;
195.     case 2:
196.         binaria(A,longi,x);
197.         break;
198.     }
199. }
200.
201. //funcion de tipo void para mostrar mensaje al usuario
202. void final(char *op2,int op){
203.     int error;
204.     do{
205.         error = 0;
206.         if (op == 3){
207.             *op2 = 'N';
208.         }
209.         else{
210.             printf("\n%cContinuar ejecuci%c? (s/n) %c ",168,162,175);
211.             scanf("%s",op2);
```

```
212.     }
213.     if (*op2 == 's' || *op2 == 'S'){
214.         system("cls");
215.     }
216.     else{
217.         if (*op2 == 'n' || *op2 == 'N'){
218.             system("cls");
219.             printf("\n\n\tGRACIAS POR USAR NUESTROS SERVICIOS");
220.             printf("\n\n\t    Equipo 8\t-\tBUAP");
221.         }
222.         else{
223.             printf("\n[!] ERROR: Ingresas una opción válida [!]",162
224. );
225.             error = 1;
226.         }
227.     } while (error!=0);
228. }
```

EJECUCIÓN:



```
C:\Users\Javier Ruiz\Desktop\Programaci3n\algoritmos\busqueda.exe

»»»Ingrese el valor x a buscar dentro del Arreglo: 80

Arreglo:
[3][17][21][36][39][51][80]

Numero 80 encontrado en posicion {6}

¿Continuar ejecución? (s/n) » _
```

```
C:\Users\Javier Ruiz\Desktop\Programaci34n\algoritmos\busqueda.exe

| ALGORITMOS DE BÚSQUEDA |

MÉTODO DE BÚSQUEDA: BINARIO

»Tamaño del Arreglo: 7

»»Proceda a ingresar los valores en el Arreglo««

Valor #1 en el Arreglo: 51
Valor #2 en el Arreglo: 21
Valor #3 en el Arreglo: 3
Valor #4 en el Arreglo: 39
Valor #5 en el Arreglo: 80
Valor #6 en el Arreglo: 36
Valor #7 en el Arreglo: 17_
```

```
C:\Users\Javier Ruiz\Desktop\Programaci34n\algoritmos\busqueda.exe

»»Ingrese el valor x a buscar dentro del Arreglo: 39

Arreglo:
[51][21][3][39][80][36][17]

Numero 39 encontrado en la posicion {3}

¿Continuar ejecución? (s/n) »
```



```
C:\Users\Javier Ruiz\Desktop\Programaci3n\algoritmos\busqueda.exe

|| ALGORITMOS DE BÚSQUEDA ||

MÉTODO DE BÚSQUEDA: LINEAL

Tamaño del Arreglo: 7

»Proceda a ingresar los valores en el Arreglo««

Valor #1 en el Arreglo: 51
Valor #2 en el Arreglo: 21
Valor #3 en el Arreglo: 3
Valor #4 en el Arreglo: 39
Valor #5 en el Arreglo: 80
Valor #6 en el Arreglo: 36
Valor #7 en el Arreglo: 17_
```

```
C:\Users\Javier Ruiz\Desktop\Programaci3n\algoritmos\busqueda.exe

|| ALGORITMOS DE BÚSQUEDA ||

MÉTODO DE BÚSQUEDA: LINEAL

Tamaño del Arreglo: 7

»Proceda a ingresar los valores en el Arreglo««

Valor #1 en el Arreglo: _
```

```
C:\Users\Javier Ruiz\Desktop\Programaci34n\algoritmos\busqueda.exe

»Indique el tamaño del Arreglo: _
```

```
C:\Users\Javier Ruiz\Desktop\Programaci34n\algoritmos\busqueda.exe

[ Menú de Algoritmos ]

Seleccione el método de búsqueda con el que desea operar:

»[1] Búsqueda lineal.
»[2] Búsqueda binaria.
»[3] Salir.

=_
```

PRUEBAS DE ESCRITORIO: Para realizar la prueba de escritorio utilice la siguiente serie de datos: 51, 21, 3, 39, 80, 36, 17.

n	Elemento	i	Arreglo							Es el dato
			0	1	2	3	4	5	6	
7	3		51	21	3	39	80	36	17	
		0								No
		1								No
		2								Si
										Se detiene el ciclo

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

n	Elemento	i	Arreglo							Es el dato
			0	1	2	3	4	5	6	
7	36		51	21	3	39	80	36	17	
		0								No
		1								No
		2								No
		3								No
		4								No
		5								Si
										Se detiene el ciclo

n	Elemento	i	Arreglo							Es el dato
			0	1	2	3	4	5	6	
7	17		51	21	3	39	80	36	17	
		0								No
		1								No
		2								No
		3								No
		4								No
		5								No
		6								Si
										Se detiene el ciclo

n	x	L	R	Found	m	Arreglo							Es el dato
						0	1	2	3	4	5	6	
7	3					3	17	21	36	39	51	80	
		0	7	false	3				36				No
			3	false	1		17						No
			1	true	0	3							Si
													Se detiene el ciclo

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA  
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

n	x	L	R	Found	m	Arreglo								Es el dato
						0	1	2	3	4	5	6		
7	36					3	17	21	36	39	51	80		
		0	7	true	3				36				Si	
													Se detiene el ciclo	

n	x	L	R	Found	m	Arreglo								Es el dato
						0	1	2	3	4	5	6		
7	80					3	17	21	36	39	51	80		
		0	7	false	3				36				No	
		3		false	5						51		No	
		5		true	6							80	Si	
													Se detiene el ciclo	