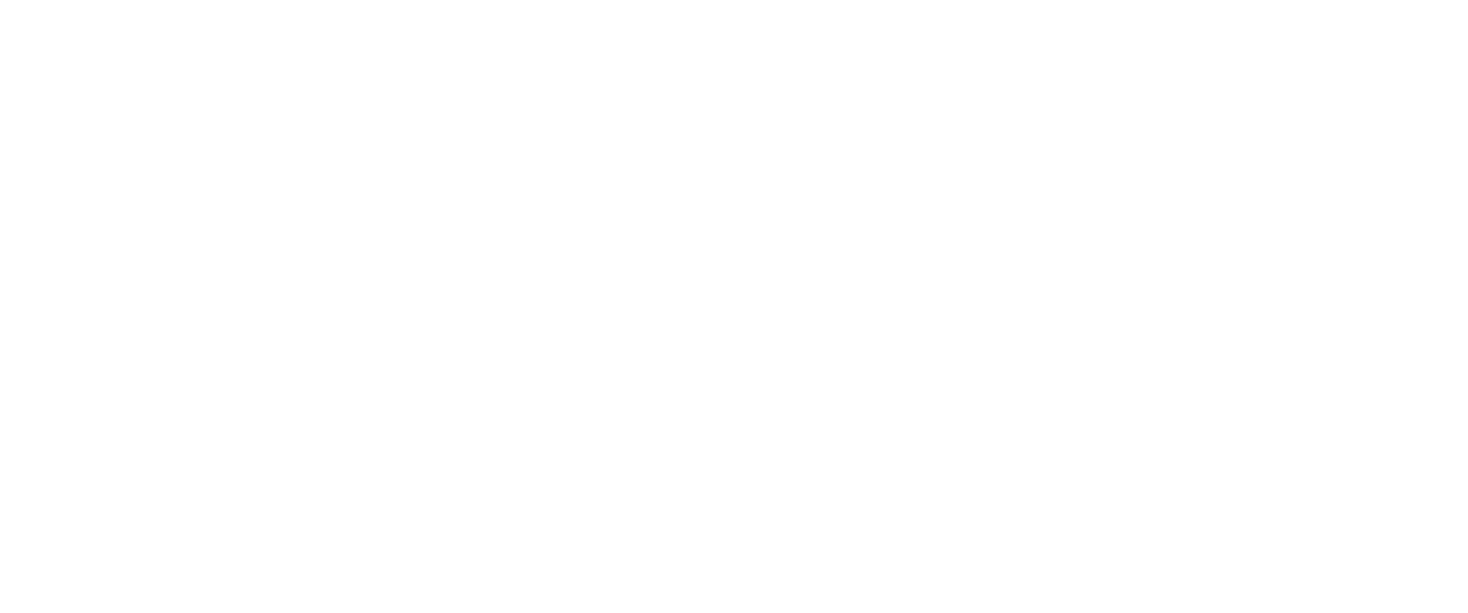
**PRACTICA 2: REPRESENTACIÓN DE INDIVIDUOS**



**PRIMER PARCIAL**

**NOMBRE:**

MARTÍNEZ ROMERO JONATHAN YAIR

**GRUPO:**

3CM5

**FECHA:**

13 DE SEPTIEMBRE DE 2018

**UNIDAD DE APRENDIZÁJE:**

ALGORITMOS GENETICOS

**PRACTICA NO.: 2**



**INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO**

**INTRODUCCIÓN**

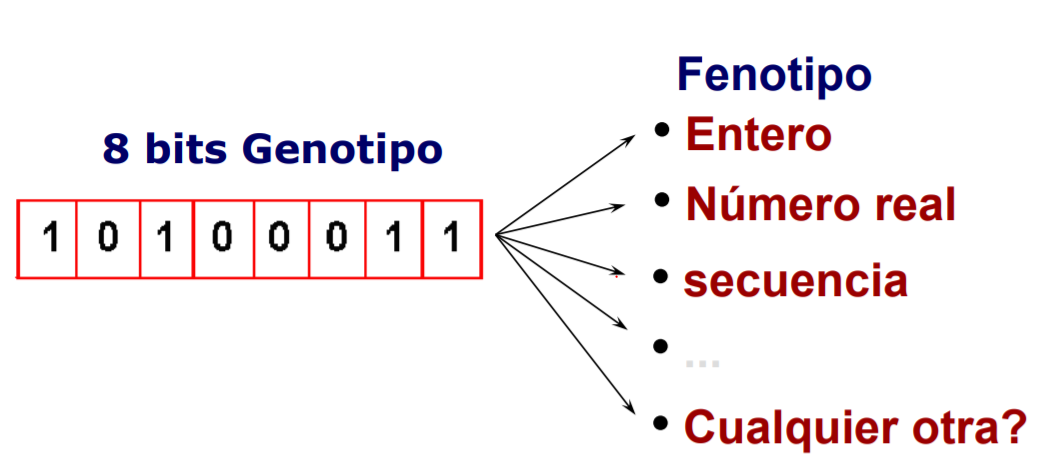
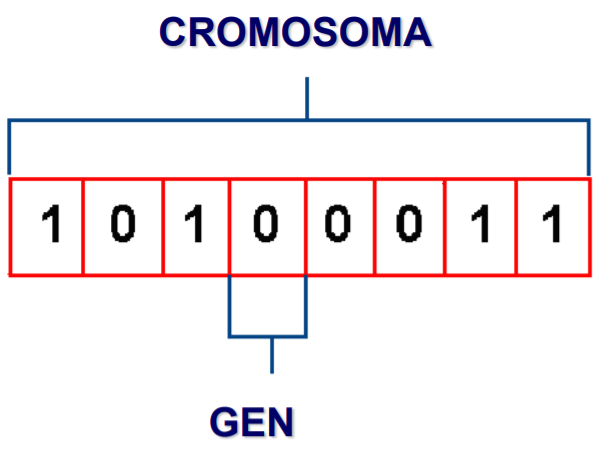
Debemos disponer de un mecanismo para codificar un individuo como un genotipo.

Existen muchas maneras de hacer esto y se hade elegir la más relevante para el problema en cuestión.

Una vez elegida una representación, hemos de tener en mente como los genotipos (codificación) serán evaluados y qué operadores genéticos hay que utilizar.

* **REPRESENTACIÓN BINARIA**

La representación de un individuo se puede hacer mediante una codificación discreta, y en particular binaria.



* **CÓDIGOS DE GRAY**

La investigación en AGs fue que el uso de la representación binaria no mapea adecuadamente el espacio de búsqueda con el espacio de representación. La codificación de Gray es parte de una familia de representaciones. Podemos convertir cualquier número binario a un código de Gray haciendo XOR a sus bits consecutivos de derecha a izquierda. Por ejemplo, dado el número101 en binario, haríamos5: 1 ⊕0 = 1, 0 ⊕1 = 1, 1 ⊕0 = 1, produciéndose (el último bit de la izquierda permanece igual) 0111, el cual es el código de Gray equivalente. Algunos investigadores han demostrado empíricamente que el uso de códigos de Gray mejora el desempeño del AG.

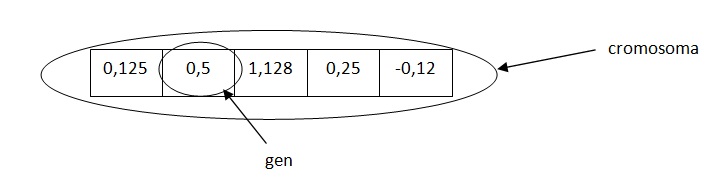
* **REPRESENTACIÓN REAL**

Una forma natural de codificar una solución es utilizando valores reales como genes

Muchas aplicaciones tienen esta forma natural de codificación

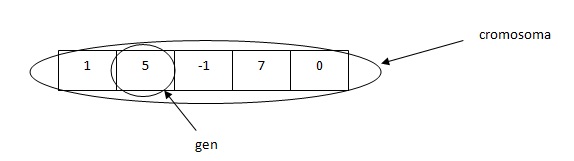
Los individuos se representan como vectores de valores reales:

La función de evaluación asocia a un vector un valor real de evaluación:



* **REPRESENTACIÓN ENTERA**

En ella se utiliza un vector cuya longitud es la del número de genes de cada individuo y el valor que puede tomar cada elemento es un número entero.



**PLANTEAMIENTO DEL PROBLEMA**

**Instrucciones.**

Realizar un programa con 4 tipos de representación de individuos, que se explicaron en clase. En lenguaje C o C++.

2.La representación binaria debe llenar un arreglo o matriz con 10 individuos, cada alelo se crea de manera aleatoria. Y así con cada una de las representaciones.

3.En total deben ser 4 programas o bien uno con un menú por representación.

**DESARROLLO**

Para la codificación se ocupó el lenguaje de Programación C mediante la utilización del IDE de Dev++.

Para su realización se hizo mediante un menú en el cual era posible elegir el algoritmo que deseábamos implementar de esta manera; para que se realizara correctamente fue necesario el uso y la implementación de números aleatorios que pudieran crear una matriz bidimensional en la que se representaron a 10 individuos es decir 10 cromosomas con 7 alelos cada uno; de esta manera se pudo crear una población con la cual se va a poder trabajar posteriormente conforme avance el curso.

**CODIFICACIÓN**

1. #include <stdio.h>
2. #include <wchar.h>
3. #include <locale.h>
4. #include <stdlib.h>
5. **#include <time.h>**
6. #include <math.h>
8. void ImprimeCromosoma(intAlelo[][7])
9. {
10. **for(int i=0;i<10;i++)**
11. {
12. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Individuo %d**\t**",i+1);
13. for(int j=0;j<7;j++)
14. {
15. **[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d,",Alelo[i][j]);**
16. }
17. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
19. }
21. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
22. }
24. void ImprimeCromosomaGray(int Alelo[][7],int AleloFinal[][7],int Decimales[])
25. **{**
26. for(int i=0;i<10;i++)
27. {
28. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Individuo %d **\t**(%d)**\t**",i+1,Decimales[i]);
29. for(int j=0;j<7;j++)
30. **{**
31. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d",Alelo[i][j]);
32. }
33. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("=>");
34. for(int j=0;j<7;j++)
35. **{**
36. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d",AleloFinal[i][j]);
37. }
38. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
40. **}**
42. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
43. }
45. **void ImprimeCromosoma(float Alelo[][7])**
46. {
47. for(int i=0;i<10;i++)
48. {
49. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Individuo %d**\t**",i+1);
50. **for(int j=0;j<7;j++)**
51. {
52. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%.2lf,",Alelo[i][j]);
53. }
54. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
56. }
58. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
59. }

62. void binaria()
63. {
64. srand(time(NULL));
66. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**Representación binaria**\n**");
67. int Alelo[10][7];
69. for(int i=0;i<10;i++)
70. **{**
71. for(int j=0;j<7;j++)
72. {
73. Alelo[i][j]=rand() % 1001;
74. if(Alelo[i][j]%2)
75. **{**
76. Alelo[i][j] = 1;
77. }
78. else
79. {
80. **Alelo[i][j] = 0;**
81. }
82. }
83. }

86. ImprimeCromosoma(Alelo);
87. }
89. void gray()
90. **{**
91. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**Códigos de Gray**\n**");
93. srand(time(NULL));
95. **int Alelo[10][7];**
96. int AleloFinal[10][7];
97. int Decimal[10];
99. for(int i=0;i<10;i++)
100. **{**
101. for(int j=0;j<7;j++)
102. {
103. Alelo[i][j]=rand() % 1001;
104. if(Alelo[i][j]%2)
105. **{**
106. Alelo[i][j] = 1;
107. }
108. else
109. {
110. **Alelo[i][j] =0;**
111. }
112. }
113. }
115. ***//ImprimeCromosoma(Alelo);***
117. int temp = 0;
118. int td = 0;
120. **for(int i=0;i<10;i++)**
121. {
123. td = 0;
125. **for(int k=0;k<7;k++)**
126. {
127. if(k == 0)
128. {
129. AleloFinal[i][k] = Alelo[i][k];
130. **}**
131. else
132. {
133. temp = k-1;
134. AleloFinal[i][k] = Alelo[i][k]^Alelo[i][temp];
135. **}**


139. }
141. for(int k=0;k<7;k++)
142. {
143. td += Alelo[i][6-k]\*pow(2,k);

146. }
148. Decimal[i] = td;
150. **}**
152. ImprimeCromosomaGray(Alelo,AleloFinal,Decimal);
154. }
156. void reales()
157. {
158. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**Codificación en números reales**\n**");
160. **srand(time(NULL));**
162. float Alelo[10][7];
164. for(int i=0;i<10;i++)
165. **{**
166. for(int j=0;j<7;j++)
167. {
168. Alelo[i][j]=(double)rand()/(RAND\_MAX+1)\*(99)+1;
170. **}**
171. }

174. ImprimeCromosoma(Alelo);

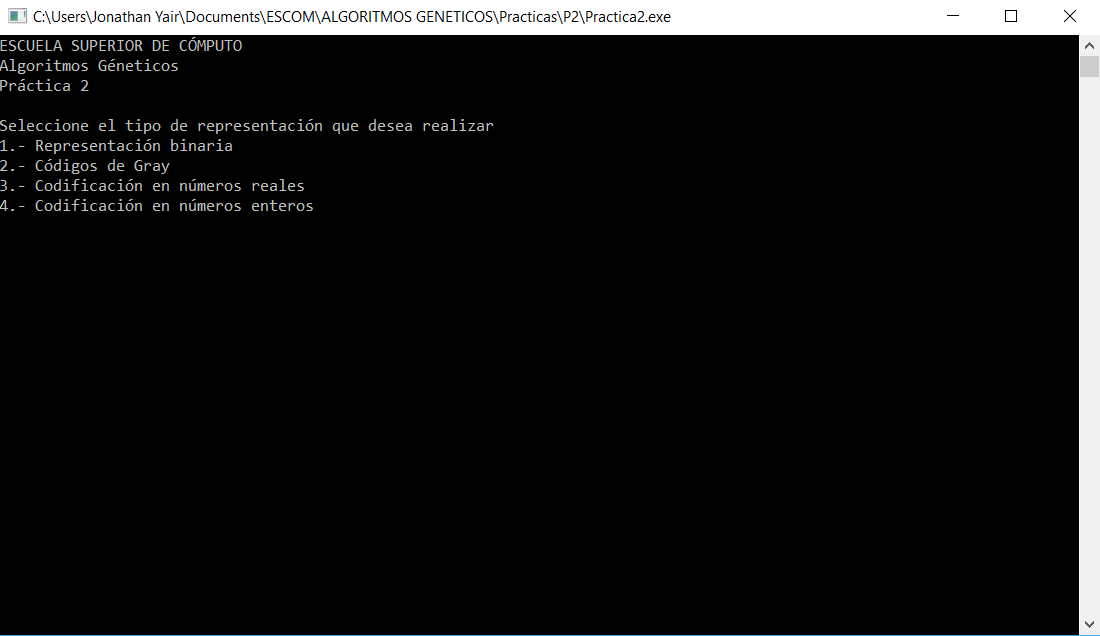
177. }
179. void enteros()
180. **{**
181. srand(time(NULL));
183. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**Codificación en números enteros**\n**");
184. int Alelo[10][7];
186. for(int i=0;i<10;i++)
187. {
188. for(int j=0;j<7;j++)
189. {
190. **Alelo[i][j]=rand() % 100;**
192. }
193. }

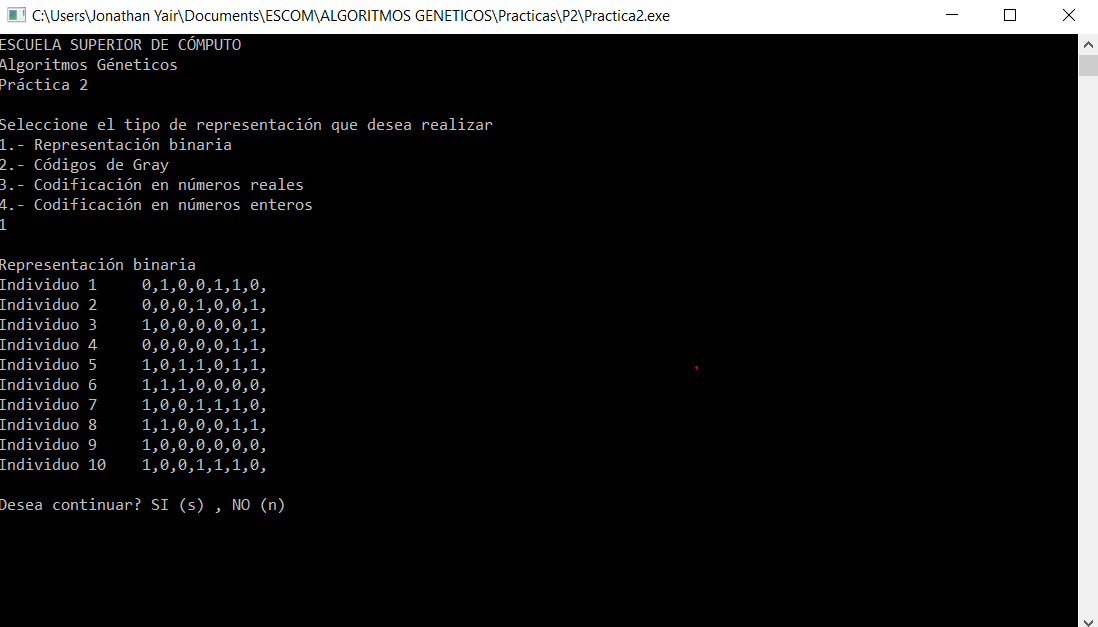
196. ImprimeCromosoma(Alelo);

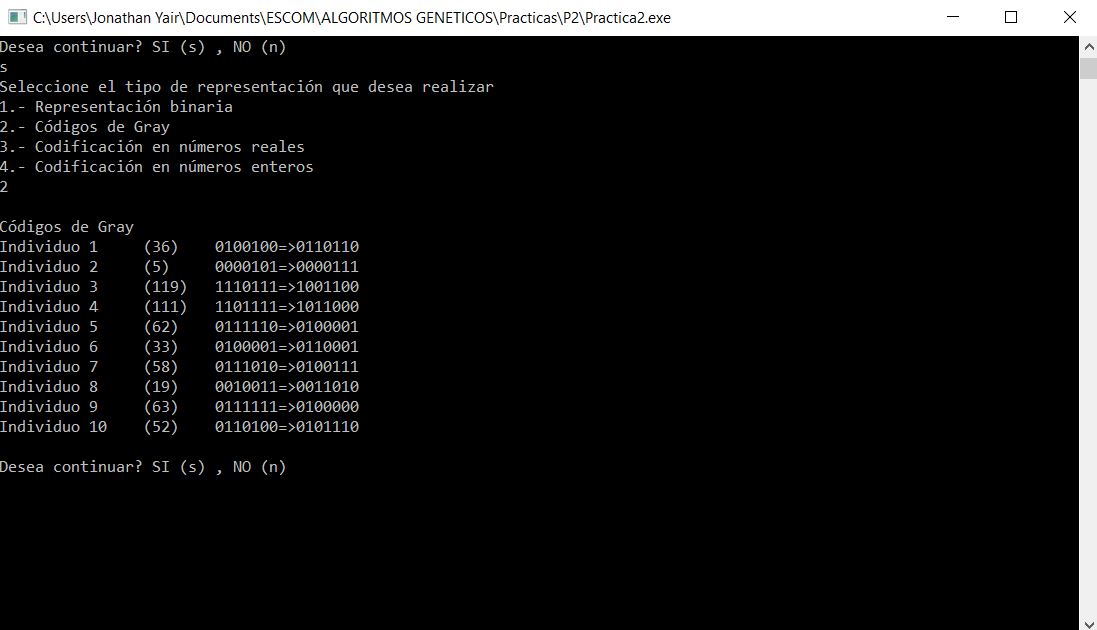
199. }
201. int main()
202. {
203. setlocale(LC\_ALL, "");
204. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("ESCUELA SUPERIOR DE CÓMPUTO**\n**");
205. [**printf**](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)**("Algoritmos Géneticos\n");**
206. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Práctica 2**\n\n**");
208. char continua = 's';
210. **while(continua == 's')**
211. {
212. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Seleccione el tipo de representación que desea realizar**\n**");
213. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("1.- Representación binaria**\n**");
214. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("2.- Códigos de Gray**\n**");
215. **[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("3.- Codificación en números reales\n");**
216. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("4.- Codificación en números enteros**\n**");
217. int opc = 0;

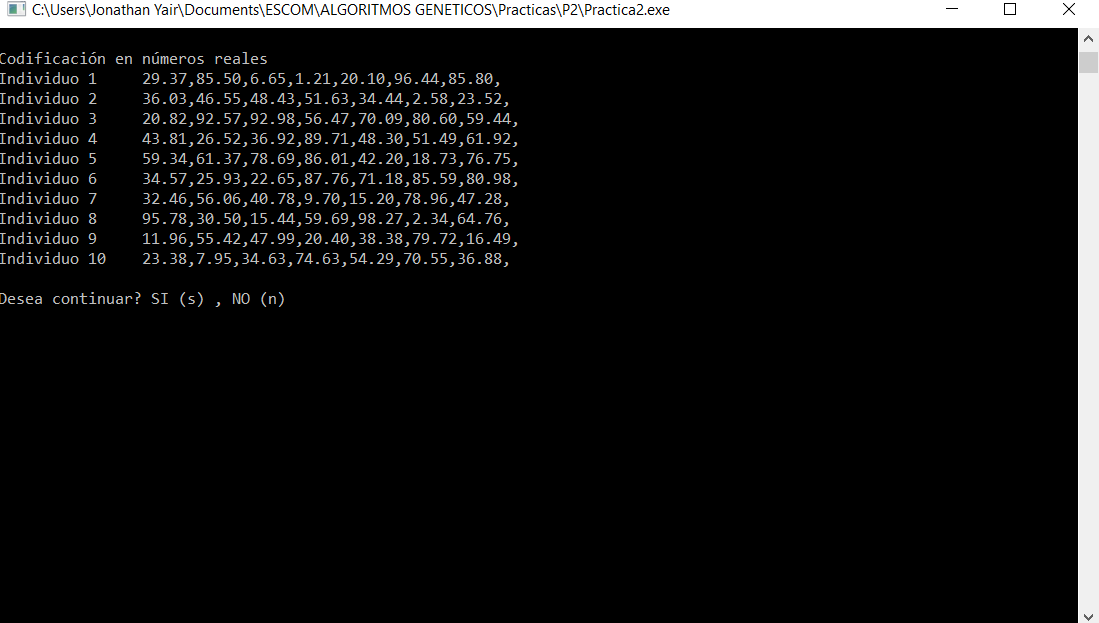
220. **scanf("%d",&opc);**
222. switch(opc)
223. {
224. case 1:
225. **binaria();**
226. **break**;
227. case 2:
228. gray();
229. **break**;
230. **case 3:**
231. reales();
232. **break**;
233. case 4:
234. enteros();
235. **break;**
236. default:
237. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("No es una opción valida**\n**");
238. **break**;
239. }
241. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Desea continuar? SI (s) , NO (n)**\n**");
242. fflush( stdin );
243. scanf("%c",&continua);
244. }
245. **return 0;**
246. }

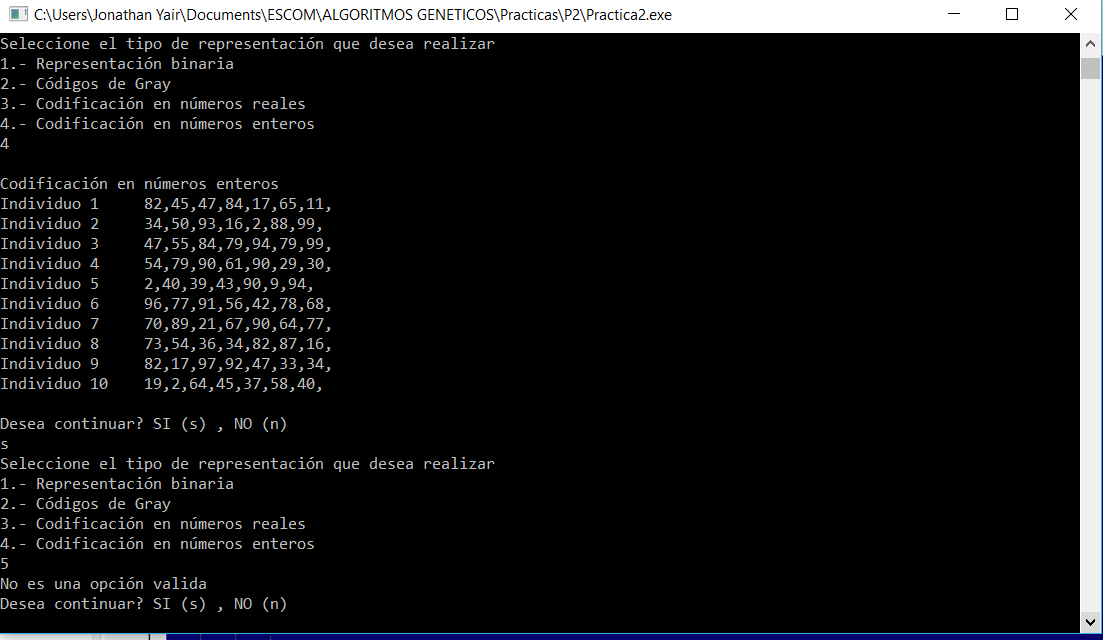
**PANTALLA**











**CONCUSIONES**

Una de las cosas que pude observar en la realización de la presenta practica es en la importancia en el uso de múltiples representaciones las cuales pueden ser usadas dependiendo el problema que estemos tratando así como los datos con los cuales se esta trabajando, de esta forma podemos enriquecer nuestro conocimiento en cuanto a la manera correcta de usar las distintas poblaciones.

En mi caso particular considero que la representación decimal y la de Gray son las más útiles ya que de esta forma hay una mayor probabilidad de que los distintos elementos salgan distintos evitando que las operaciones coincidan en el caso de diversos algoritmos utilizados más adelante.