INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Genetic Algorithms

Practica 3:

Profa.: MORALES GUITRON SANDRA LUZ

Grupo: 3CM5

Alumno:

Salcedo Barrón Ruben Osmair.

MEXICO, D.F. a 11 de octubre del 2018

**Introducción**

Propuesto por DeJong, es posiblemente el método más utilizado desde los orígenes de los Algoritmos Genéticos [Blickle and Thiele, 1995].

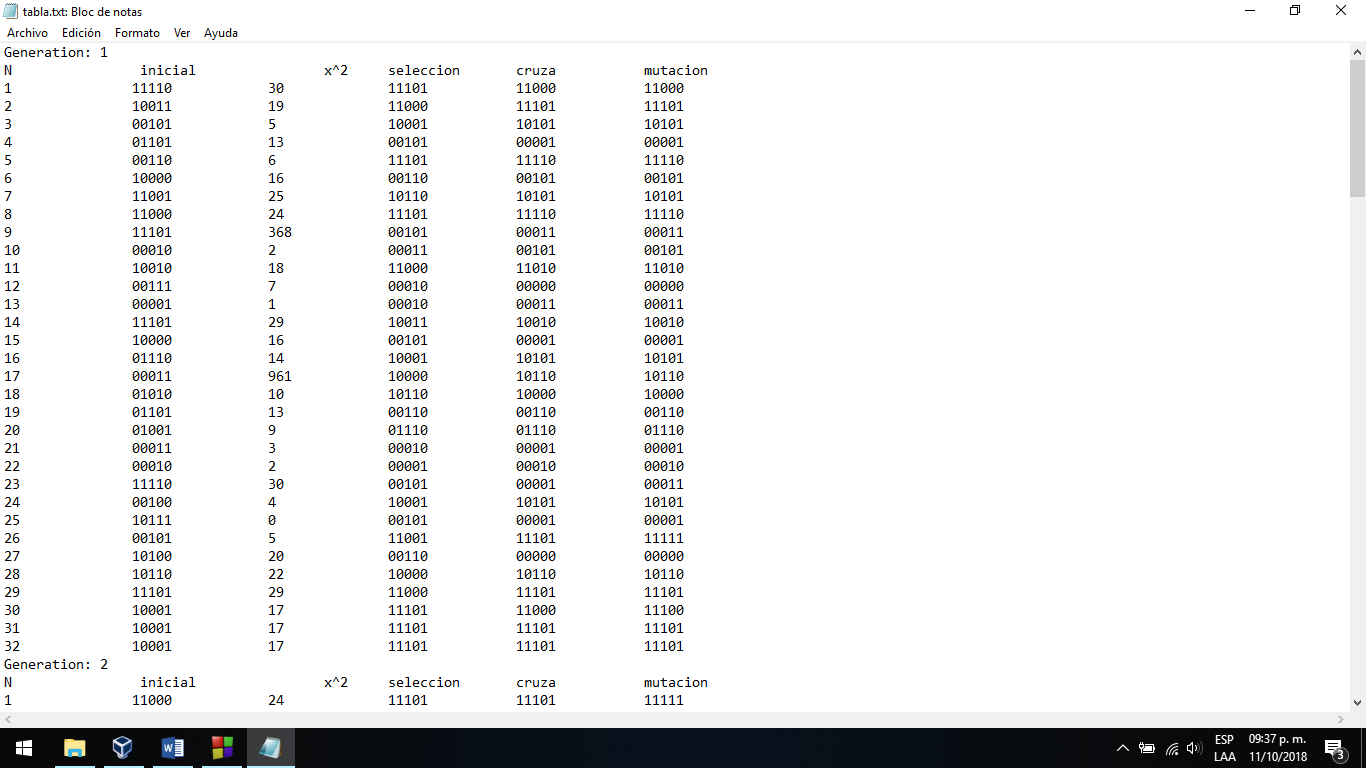
A cada uno de los individuos de la población se le asigna una parte proporcional a su ajuste de una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente la población está ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio del intervalo [0...1] y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido.

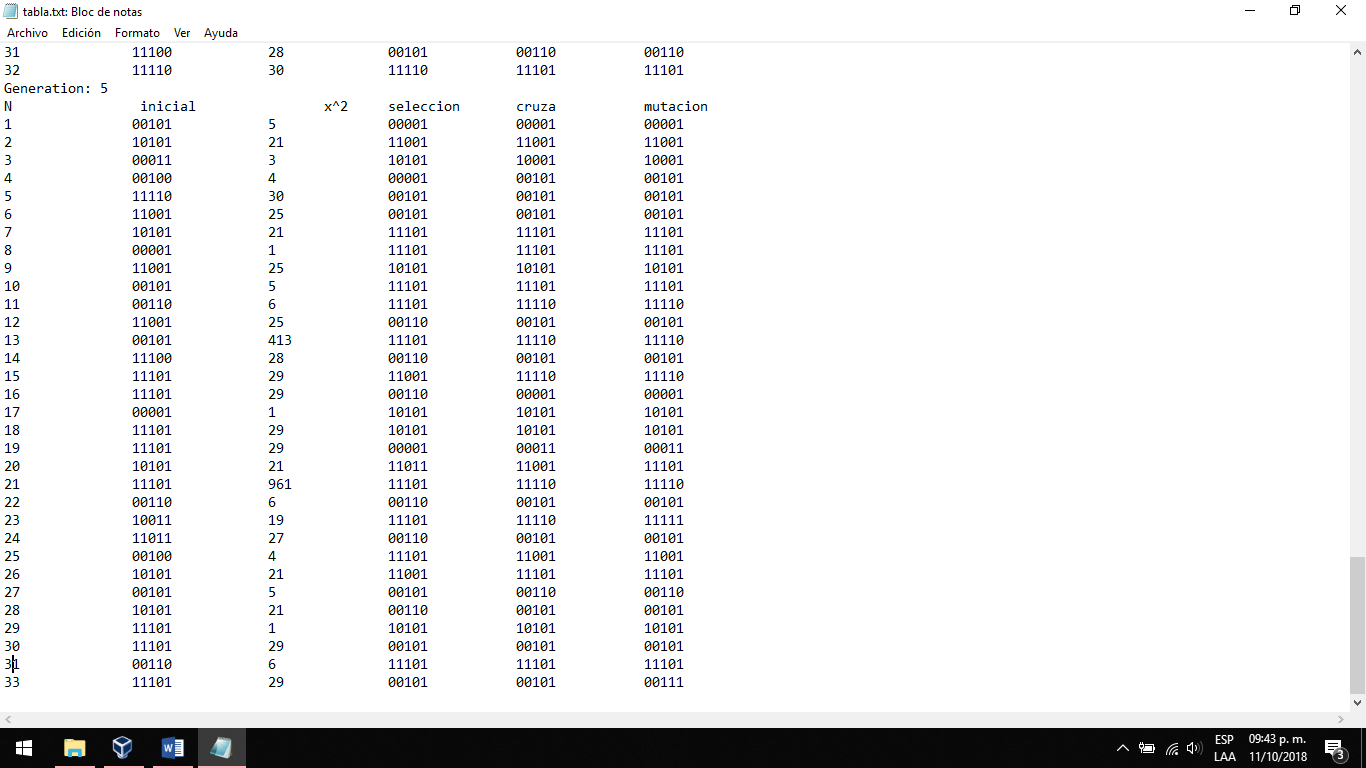
Es un método muy sencillo, pero ineficiente a medida que aumenta el tamaño de la población (su complejidad es O(n^2). Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez.

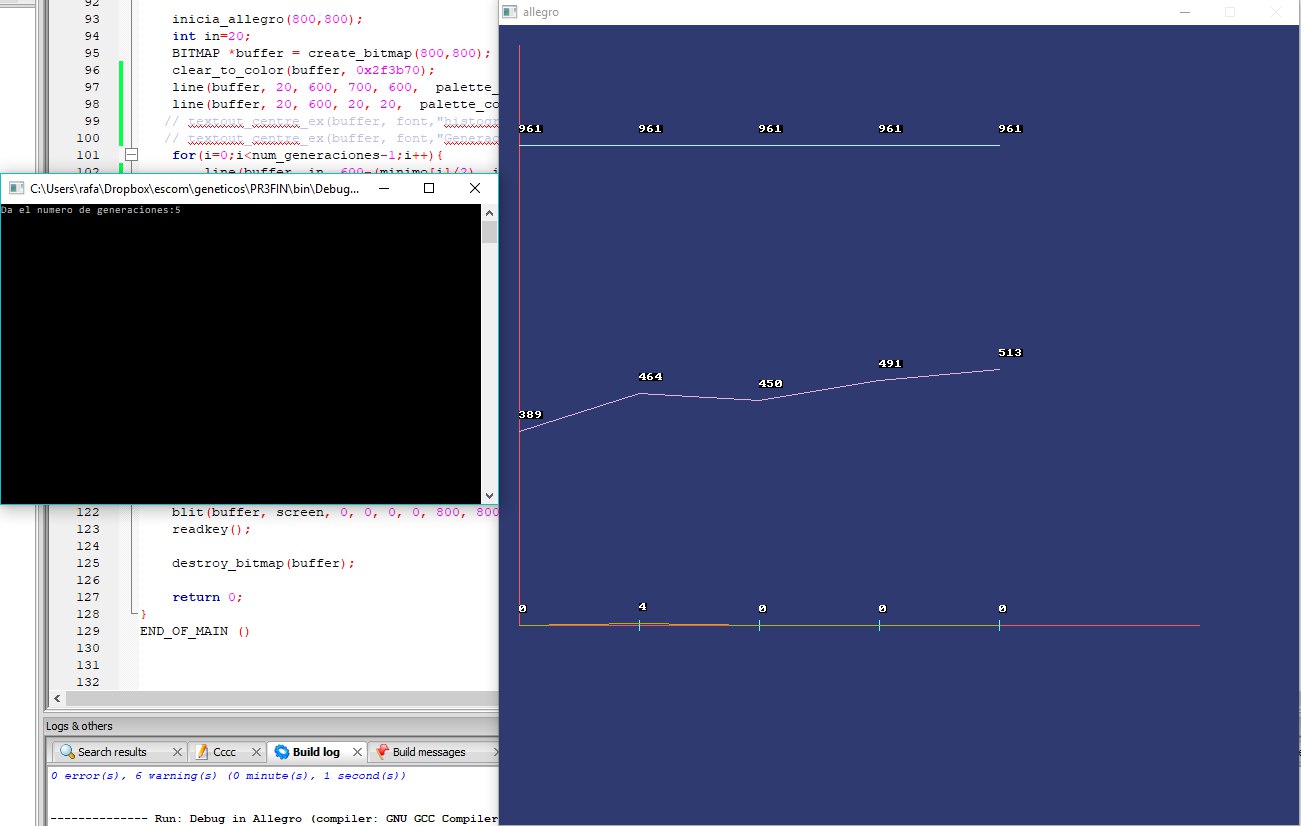
**Desarrollo**

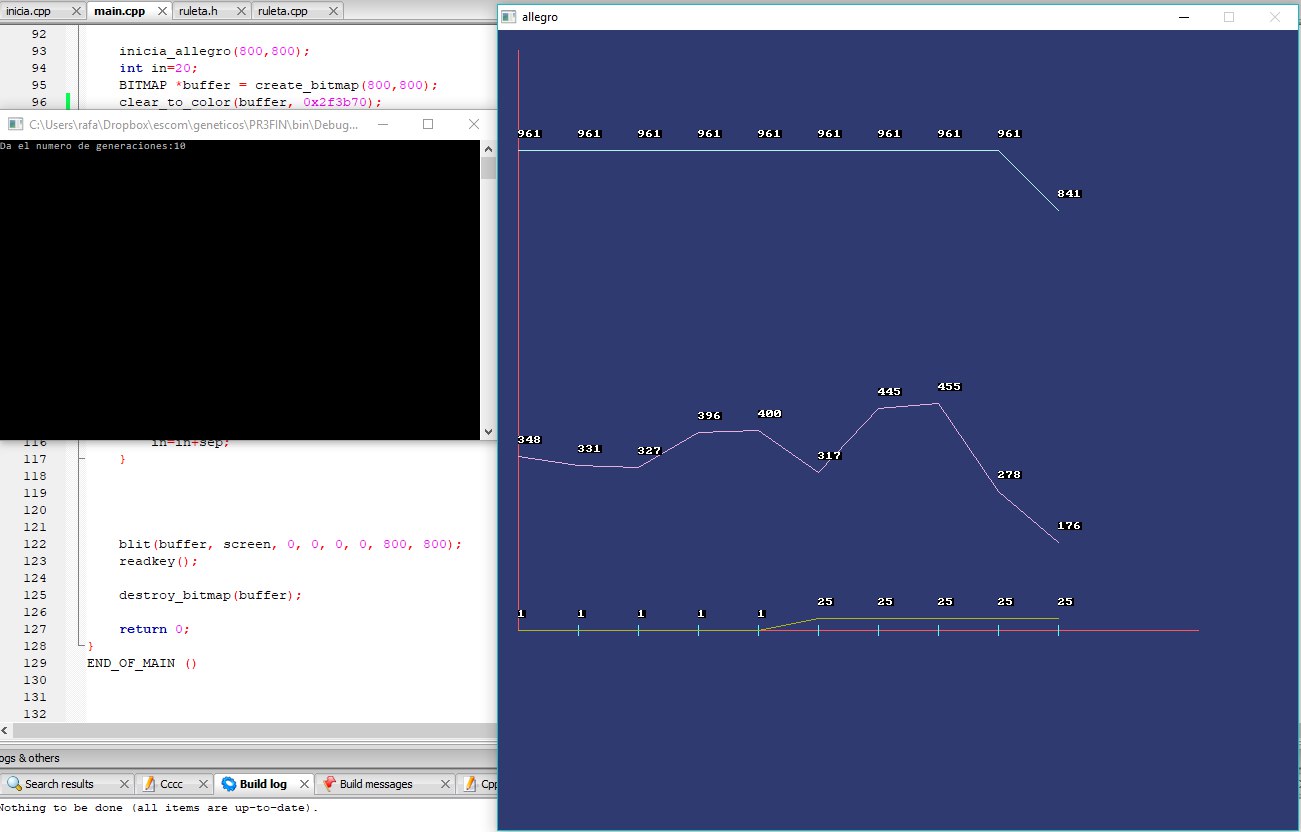
En este algoritmo de ruleta, implementamos 4 arreglos uno para población inicial, de individuos seleccionados por ruleta, de cruza y de mutación. Primero se llena aleatoriamente el arreglo de población inicial con series de 5 bits después se hace el algoritmo de ruleta en este se genera un numero aleatorio entre 0 y el valor de la aptitud total de la población inicial, después se acumula la actitud de la población y este se detiene cuando supera el valor aleatorio generado. Por ultimo el padre será el individuo que genere que se sobrepase el valor aleatorio generado.

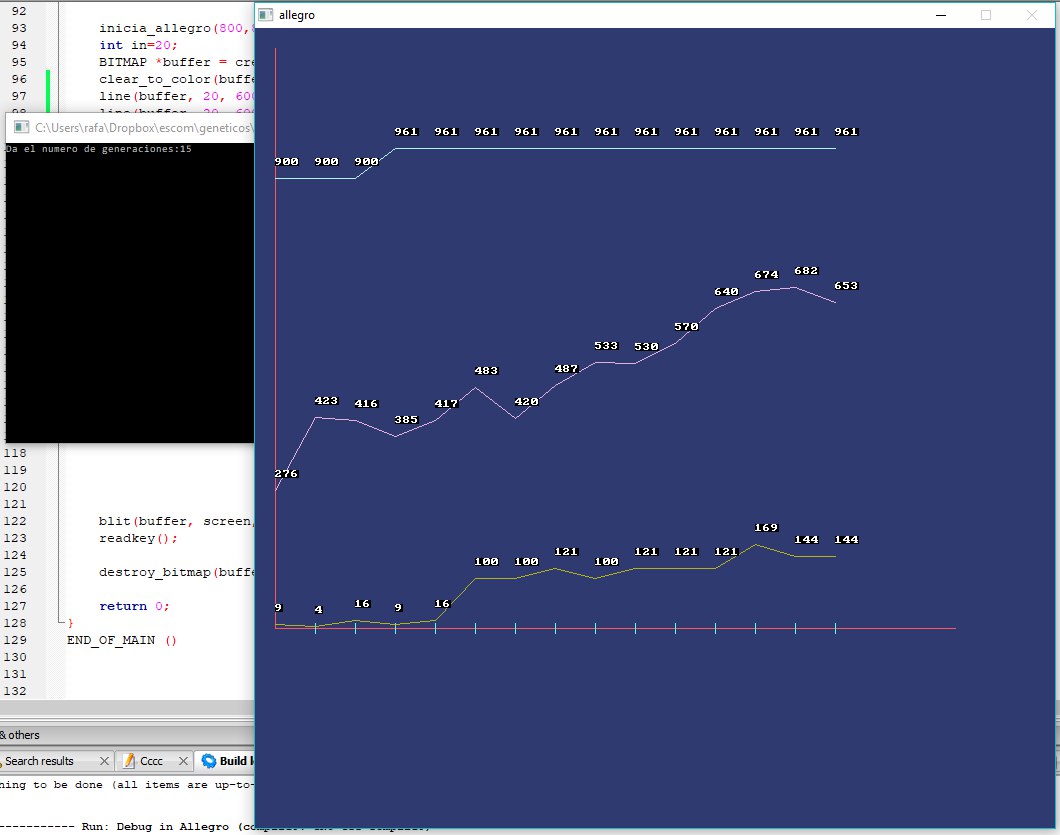
Una vez teniendo la población de selección de padres, se realiza una cruza de individuos y al obtener la población de individuos después de la cruza se necesita realizar una mutación. Para este caso será una mutación del 10 % de la población. Nuestra población total es de 32 elementos. Una vez que se tenga la población mutada, se establece esta como población inicial, para realizar el algoritmo de ruleta en la siguiente generación.











**Conclusiones**

En esta practica pudimos ver la evolución de los individuos mediante el algoritmo de ruleta, pero debido a que es aleatorio esta selección puede que en algunos casos el individuo seleccionado sea el que tenga menor aptitud con respecto a los otros. Sin embargo, el uso de este algoritmo nos puede ayudar a ver como seria la selección de la naturaleza que no siempre el mas apto es el seleccionado.

**Código**

main.cpp

#include <allegro.h>

#include "inicia.h"

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <fstream>

#include <bitset>

#include "ruleta.h"

**using** **namespace** std**;**

int main **()**

**{**

int k**=**0**,**i**=**0**;**

ofstream tabla**;**

tabla**.**open**(**"resultado.txt"**,** fstream**::**out**);**

int num\_generaciones**;**

printf**(**"Cuantas generaciones:"**);**

scanf**(**"%d"**,&**num\_generaciones**);**

bitset**<**BIT\_IND**>** inicial**[**NUM\_IND**];**

bitset**<**BIT\_IND**>** seleccion**[**NUM\_IND**];**

bitset**<**BIT\_IND**>** cruza**[**NUM\_IND**];**

bitset**<**BIT\_IND**>** mutacion\_des**[**NUM\_IND**];**

int minimo**[**num\_generaciones**];**

int maximo**[**num\_generaciones**];**

int generationValuesAverage**[**num\_generaciones**];**

int ind\_apti**[**NUM\_IND**];**

IniciarInd**(**inicial**);**

**for** **(** k **=** 0**;** k **<** num\_generaciones**;** k**++){**

int totalAptitude **=** getTotalAptitude**(**inicial**);**

**for(**i**=**0**;**i**<**NUM\_IND**;**i**++){**

ind\_apti**[**i**]=(**int**)**getIndividualAptitude**(**inicial**[**i**]);**

**}**

**for** **(** i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

seleccion**[**i**]** **=** rouletteSelection**(**inicial**,** totalAptitude**);**

**}**

**for** **(** i **=** 0**;** i **<** NUM\_IND**;** i**+=**2**)**

**{**

cruza**[**i**]** **=** crossAlgorithm**(**seleccion**[**i**],** seleccion**[**i **+** 1**]);**

cruza**[**i **+** 1**]** **=** crossAlgorithm**(**seleccion**[**i **+** 1**],** seleccion**[**i**]);**

**}**

**for** **(** i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

mutacion\_des**[**i**]** **=** cruza**[**i**];**

**}**

int mutation\_value **=** NUM\_IND **/** PROBABILITY**;**

srand **(**time**(NULL));**

**for** **(**i **=** 0**;** i **<** mutation\_value**;** i**++)**

**{**

int indivual\_to\_mutate **=** rand**()** **%** NUM\_IND**;**

mutacion\_des**[**indivual\_to\_mutate**]** **=** mutationAlgorithm**(**cruza**[**indivual\_to\_mutate**]);**

**}**

int min\_gen\_value **=** getMinGenerationValue**(**mutacion\_des**);**

int max\_gen\_value **=** getMaxGenerationValue**(**mutacion\_des**);**

int gen\_average **=** getGenerationAverage**(**mutacion\_des**);**

minimo**[**k**]** **=** min\_gen\_value**;**

maximo**[**k**]** **=** max\_gen\_value**;**

generationValuesAverage**[**k**]** **=** gen\_average**;**

tabla **<<** "Generation: " **<<** k**+**1 **<<** endl**;**

tabla **<<** "N\t\t inicial\t\tx^2\t\tseleccion\t\tcruza\t\tmutacion\t\t"**<<**endl**;**

int indice**=**1**;**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

tabla **<<** indice **<<**"\t\t"**<<** inicial**[**i**]** **<<** "\t\t\t"**<<**ind\_apti**[**i**]<<**"\t\t\t" **<<** seleccion**[**i**]** **<<** "\t\t\t"**<<** cruza**[**i**]** **<<** "\t\t\t" **<<** mutacion\_des **[**i**]** **<<** endl**;**

indice**+**1**;**

**}**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

inicial**[**i**]** **=** mutacion\_des**[**i**];**

**}**

**}**

int sep**=**600**/**num\_generaciones**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

inicia\_allegro**(**800**,**800**);**

int in**=**20**;**

BITMAP **\***buffer **=** create\_bitmap**(**800**,**800**);**

clear\_to\_color**(**buffer**,** 0x0a6c92**);**

line**(**buffer**,** 20**,** 600**,** 700**,** 600**,** palette\_color**[**11**]);**

line**(**buffer**,** 20**,** 600**,** 20**,** 20**,** palette\_color**[**11**]);**

textout\_centre\_ex**(**buffer**,** font**,**"histograma"**,** 160**,** 25**,** 0xFFFFFF**,** 0x999999**);**

textout\_centre\_ex**(**buffer**,** font**,**"Generaciones"**,** 160**,** 650**,** 0xFFFFFF**,** 0x999999**);**

**for(**i**=**0**;**i**<**num\_generaciones**-**1**;**i**++){**

line**(**buffer**,** in**,** 600**-(**minimo**[**i**]/**2**),** in**+**sep**,** 600**-(**minimo**[**i**+**1**]/**2**),** 0xbde4ff**);**

line**(**buffer**,** in**,** 600**-(**maximo**[**i**]/**2**),** in**+**sep**,** 600**-(**maximo**[**i**+**1**]/**2**),** 0xe5ffdc**);**

line**(**buffer**,** in**,** 600**-(**generationValuesAverage**[**i**]/**2**),** in**+**sep**,** 600**-(**generationValuesAverage**[**i**+**1**]/**2**),** 0xe5b0dc**);**

line**(**buffer**,** in**+**sep**,** 605**,** in**+**sep**,** 595**,** palette\_color**[**11**]);**

in**=**in**+**sep**;**

**}**

in**=**20**;**

**for(**i**=**0**;**i**<**num\_generaciones**;**i**++){**

textprintf**(**buffer**,** font**,** in**,** 580**-(**minimo**[**i**]/**2**),** 0xFFFFFF**,** "%d"**,(**minimo**[**i**]));**

textprintf**(**buffer**,** font**,** in**,** 580**-(**maximo**[**i**]/**2**),** 0xFFFFFF**,** "%d"**,(**maximo**[**i**]));**

textprintf**(**buffer**,** font**,** in**,** 580**-(**generationValuesAverage**[**i**]/**2**),** 0xFFFFFF**,** "%d"**,(**generationValuesAverage**[**i**]));**

textprintf**(**buffer**,** font**,** in**,** 610**,** 0xFFFFFF**,** "%d"**,(**i**));**

in**=**in**+**sep**;**

**}**

blit**(**buffer**,** screen**,** 0**,** 0**,** 0**,** 0**,** 800**,** 800**);**

readkey**();**

destroy\_bitmap**(**buffer**);**

**return** 0**;**

**}**

END\_OF\_MAIN **()**

ruleta.cpp

#include "ruleta.h"

#include <iostream>

#include <time.h>

**using** **namespace** std**;**

void IniciarInd**(**bitset**<**BIT\_IND**>** array**[]){**

srand **(**time**(NULL));**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

array**[**i**]** **=** 1**+**rand**()%(**101**-**1**);**

**}**

void printIndividuals**(**bitset**<**BIT\_IND**>** array**[]){**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

cout **<<** array**[**i**].**to\_ulong**()** **<<** endl**;**

**}**

int getIndividualValue**(**bitset**<**BIT\_IND**>** individual**){**

**return** individual**.**to\_ulong**();**

**}**

int getIndividualAptitude**(**bitset**<**BIT\_IND**>** individual**){**

**return** **(**individual**.**to\_ulong**()\***individual**.**to\_ulong**());**

**}**

int getTotalAptitude**(**bitset**<**BIT\_IND**>** array**[]){**

int total **=** 0**;**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

total **+=** getIndividualAptitude**(**array**[**i**]);**

**}**

**return** total**;**

**}**

bitset**<**BIT\_IND**>** rouletteSelection**(**bitset**<**BIT\_IND**>** array**[],** int totalAptitude**)** **{**

int r **=** rand**()** **%** **(**totalAptitude **+** 1**);**

int add **=** 0**;**

int i**;**

**for(**i **=** 0**;** i **<** NUM\_IND **&&** add **<** r**;** i**++)** **{**

add **+=** getIndividualAptitude**(**array**[**i**]);**

**}**

**return** array**[**i**];**

**}**

bitset**<**BIT\_IND**>** crossAlgorithm**(**bitset**<**BIT\_IND**>** **&**p1**,** bitset**<**BIT\_IND**>** **&**p2**)** **{**

bitset**<**BIT\_IND**>** aux **=** p1**;**

**for** **(**int i **=** 0**;** i **<=** PUNTO\_CRUZA**;** i**++)**

**{**

aux**.**set**(**PUNTO\_CRUZA **-** i**,** p2**[**PUNTO\_CRUZA **-** i**]);**

**}**

**return** aux**;**

**}**

bitset**<**BIT\_IND**>** mutationAlgorithm**(**bitset**<**BIT\_IND**>** individual**){**

bitset**<**BIT\_IND**>** result **=** individual**;**

int cont **=** 0**;**

**while(**cont **<=** MAX\_SEARCH\_VALUE**)**

**{**

int mutation\_point **=** rand**()** **%** BIT\_IND**;**

**if(**result**[**mutation\_point**]** **==** 0**){**

result**.**set**(**mutation\_point**,** 1**);**

**break;**

**}**

cont**++;**

**}**

**return** result**;**

**}**

int getMinGenerationValue**(**bitset**<**BIT\_IND**>** array**[]){**

int min **=** 1000000**,** aux **=** 0**;**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

aux **=** getIndividualAptitude**(**array**[**i**]);**

**if(**aux **<** min**){**

min **=** aux**;**

**}**

**}**

**return** min**;**

**}**

int getMaxGenerationValue**(**bitset**<**BIT\_IND**>** array**[]){**

int max **=** 0**,** aux **=** 0**;**

**for** **(**int i **=** 0**;** i **<** NUM\_IND**;** i**++)**

**{**

aux **=** getIndividualAptitude**(**array**[**i**]);**

**if(**aux **>** max**){**

max **=** aux**;**

**}**

**}**

**return** max**;**

**}**

int getGenerationAverage**(**bitset**<**BIT\_IND**>** array**[]){**

**return** **(**getTotalAptitude**(**array**)/**NUM\_IND**);**

**}**

inicia.cpp

#include "inicia.h"

#include <allegro.h>

void inicia\_allegro**(**int ANCHO\_ **,** int ALTO\_**){**

allegro\_init**();**

install\_keyboard**();**

set\_color\_depth**(**32**);**

set\_gfx\_mode**(**GFX\_AUTODETECT\_WINDOWED**,** ANCHO\_**,** ALTO\_**,** 0**,** 0**);**

**}**

int inicia\_audio**(**int izquierda**,** int derecha**){**

**if** **(**install\_sound**(**DIGI\_AUTODETECT**,** MIDI\_AUTODETECT**,** **NULL)** **!=** 0**)** **{**

allegro\_message**(**"Error: inicializando sistema de sonido\n%s\n"**,** allegro\_error**);**

**return** 1**;**

**}**

set\_volume**(**izquierda**,** derecha**);**

**return** 0**;**

**}**