INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Genetic Algorithms

Practica 4:

Profa.: MORALES GUITRON SANDRA LUZ

Grupo: 3CM5

Alumno:

Salcedo Barrón Ruben Osmair.

MEXICO, D.F. a 1 de noviembre del 2018

**Introducción**

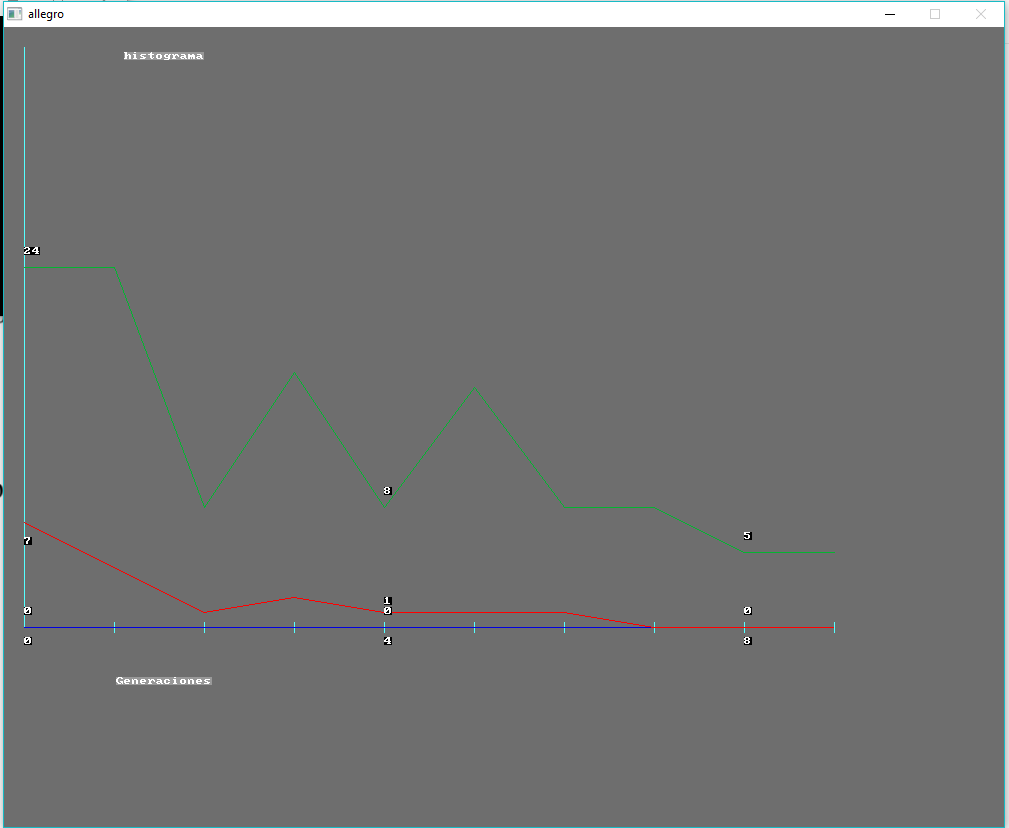
El método de selección por torneo es escoger individuos con base en comparaciones directas entre ellos.

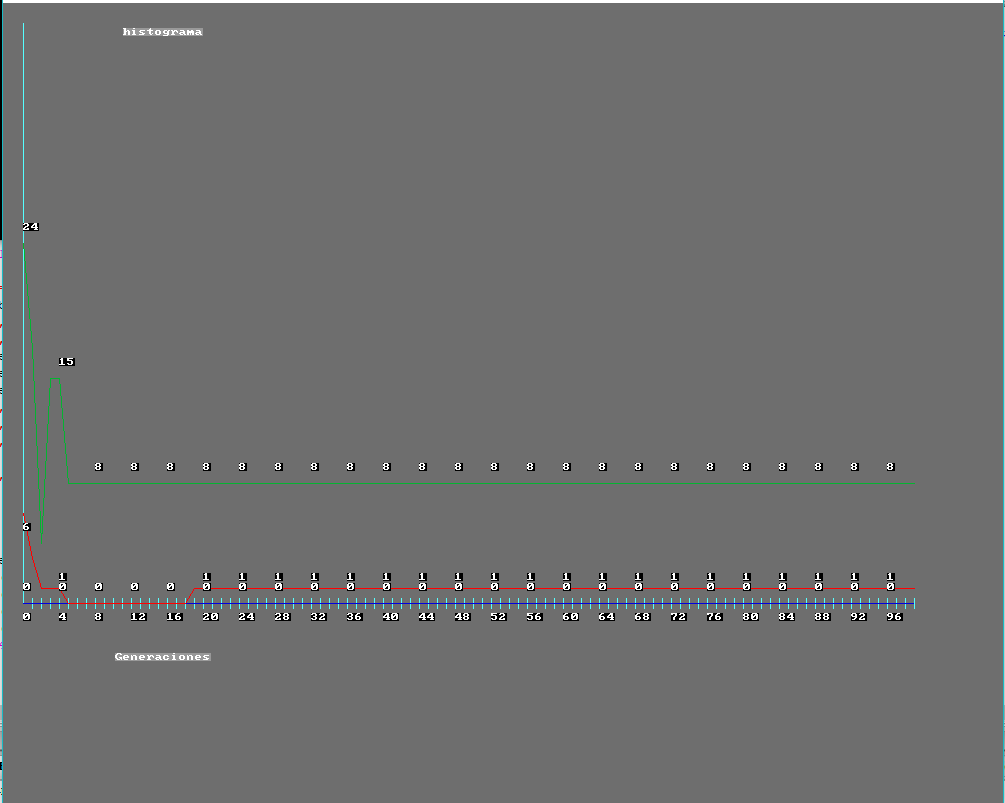
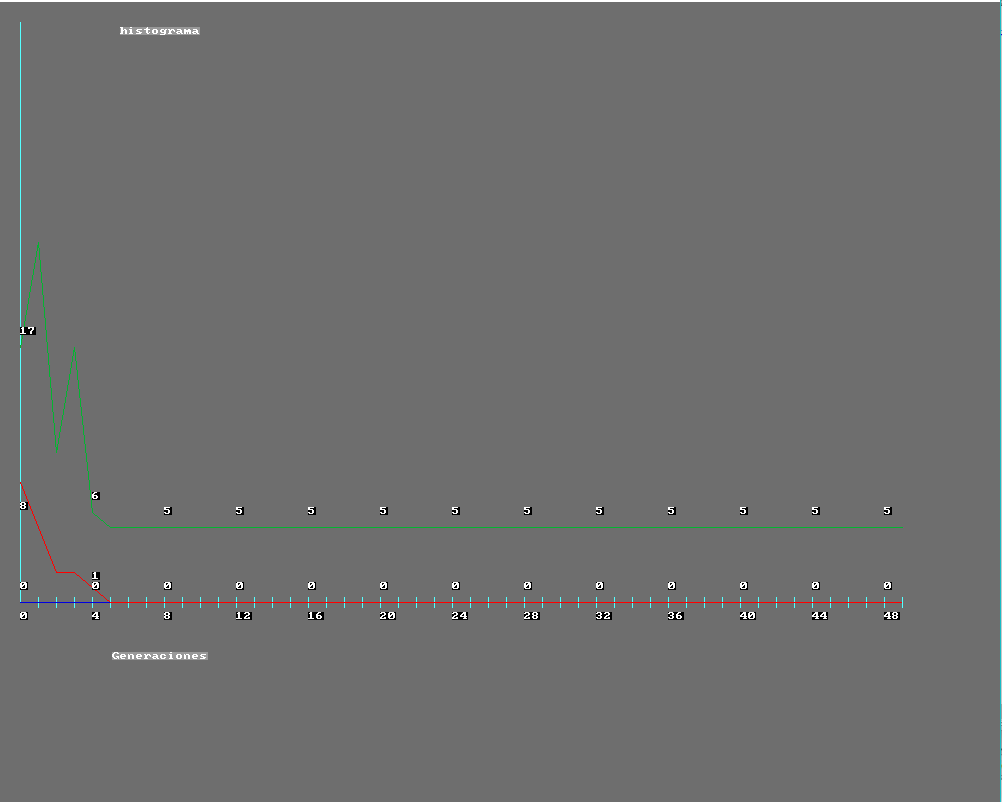
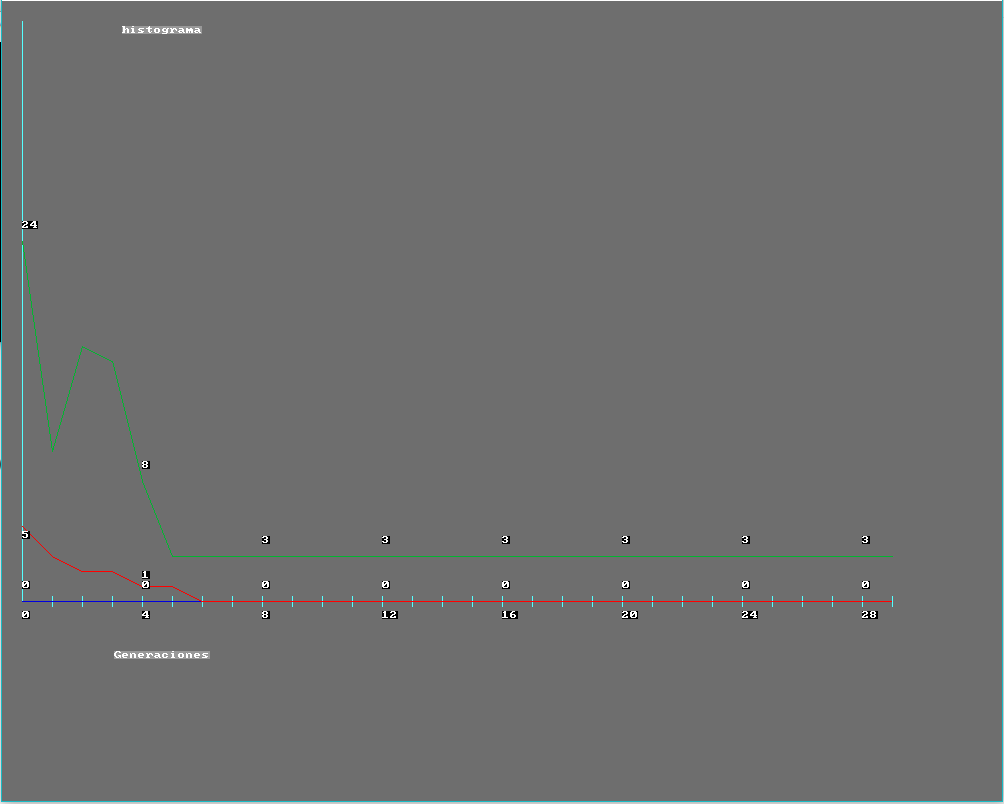
En esta práctica veremos 1 tipo de los 2 posibles de selección mediante torneo: **Determinística, Probabilística**. En esta práctica implementamos la versión probabilística, la cual consiste en barajar los individuos de la población, escoger un numero p de individuos después compararlos con base en sus aptitudes, generar un flip y por último elegir al individuo más apto si flip resulta verdadero, en caso contrario se elige al individuo menos apto.

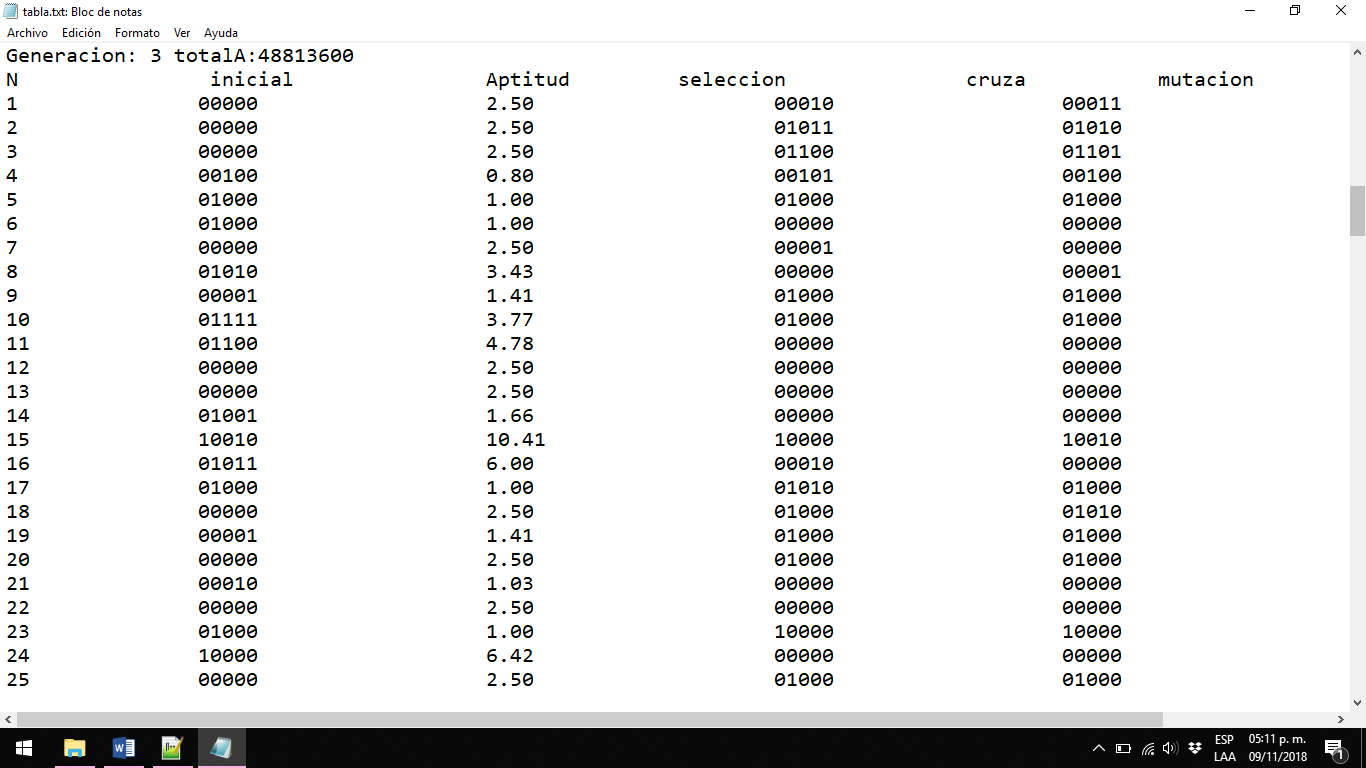
**Desarrollo**

En esta práctica se implementó 4 arreglos de bits para cada etapa en el algoritmo: Población inicial, Población de individuos seleccionados mediante el algoritmo de selección por torneo, Población después de cruza. Población después de mutación.

Al principio se llena aleatoriamente el arreglo de población inicial con series de 5 bits, después para implementar la selección por torneo, se barajean los individuos de la población para así generar parejas para el enfrentamiento. En el primer enfrentamiento, se seleccionan 16 individuos ganadores, se tiene que volver a realizar el algoritmo, para de este modo completar los 32 individuos. Una vez teniendo la población de selección de padres, se realiza una cruza de individuos. Se muestran ejemplos con 10, 30, 50 y 100 generaciones.







**Conclusiones**

La selección probabilística nos dice que los individuos pueden no ser los más aptos, a diferencia de la selección determinista, que siempre nos dará el individuo más apto. Como nos decía en la clase la naturaleza tiene varios factores para determinar un ganador y a veces este ganador no es el más apto y en esta practica pudimos simular algo similar.

**Código**

**main.cpp**

#include <allegro.h>

#include "inicia.h"

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <fstream>

#include <bitset>

#include "torneo.h"

#include <iomanip>

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

int main **()**

**{**

int k**=**0**,**i**=**0**,**h**=**0**,**w**=**0**,**f**=**0**;**

ofstream tabla**;**

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

int num\_generaciones**;**

printf**(**"Da el numero de 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**];**

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

float proba**[**NUM\_IND**];**

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

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

w**=**0**;**f**=**0**;**

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

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

**for(**h**=**0**;**h**<**NUM\_IND**/**NUM\_BARA**;**h**++){**

seleccion**[**w**]** **=** tournamentSelection**(**inicial**[**f**],** inicial**[**f**+**1**]);**

w**++;**

f**=**f**+**2**;**

**}**

swapInd**(**inicial**);**

**}**

**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**]);**

**}**

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

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

**}**

int min\_gen\_value **=** getMinGenerationValue**(**inicial**);**

int max\_gen\_value **=** getMaxGenerationValue**(**inicial**);**

int gen\_average **=** getGenerationAverage**(**inicial**);**

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

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

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

tabla **<<** "Generacion: " **<<** k**+**1 **<<**" "**<<**"totalA:"**<<** totalAptitude**<<** endl**;**

tabla **<<** "N\t\t inicial\t\tAptitud\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"**<<**std**::**fixed**<<**std**::**setprecision**(**2**)<<**ind\_apti**[**i**]<<**"\t\t\t" **<<** seleccion**[**i**]** **<<** "\t\t\t"**<<** cruza**[**i**]** **<<** "\t\t\t" **<<** mutacion\_des **[**i**]<<** "\t\t"**<<** endl**;**

indice**++;**

**}**

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

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

**}**

**}**

int sep**=**900**/**num\_generaciones**;**

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

inicia\_allegro**(**1000**,**800**);**

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

BITMAP **\***buffer **=** create\_bitmap**(**1000**,**800**);**

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

line**(**buffer**,** 20**,** 600**,** 800**,** 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**]\***15**),** in**+**sep**,** 600**-(**minimo**[**i**+**1**]\***15**),** 0xbde4ff**);**

line**(**buffer**,** in**,** 600**-(**maximo**[**i**]\***15**),** in**+**sep**,** 600**-(**maximo**[**i**+**1**]\***15**),** 0xe5ffdc**);**

line**(**buffer**,** in**,** 600**-(**generationValuesAverage**[**i**]\***15**),** in**+**sep**,** 600**-(**generationValuesAverage**[**i**+**1**]\***15**),** 0xe5b0dc**);**

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

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

**}**

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

**for(**i**=**0**;**i**<**num\_generaciones**;**i**+=**4**){**

textprintf**(**buffer**,** font**,** in**,** 580**-(**minimo**[**i**]\***15**),** 0xFFFFFF**,** "%d"**,(**minimo**[**i**]));**

textprintf**(**buffer**,** font**,** in**,** 580**-(**maximo**[**i**]\***15**),** 0xFFFFFF**,** "%d"**,(**maximo**[**i**]));**

textprintf**(**buffer**,** font**,** in**,** 580**-(**generationValuesAverage**[**i**]\***10**),** 0xFFFFFF**,** "%d"**,(**generationValuesAverage**[**i**]));**

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

in**=**in**+(**sep**\***4**);**

**}**

blit**(**buffer**,** screen**,** 0**,** 0**,** 0**,** 0**,** 1000**,** 800**);**

readkey**();**

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

**return** 0**;**

**}**

END\_OF\_MAIN **()**

**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**;**

**}**

**torneo.cpp**

#include "torneo.h"

#include <iostream>

#include <time.h>

#include <stdio.h>

#include <array>

#include <algorithm>

#include <vector>

**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**();**

**}**

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

float result**;**

float numerador**;**

float denominador**;**

float x **=** individual**.**to\_ulong**();**

float senX **=** sin**(**x**);**

numerador **=** x **-** 5**;**

denominador **=** 2 **+** senX**;**

result **=** abs**(**numerador**/**denominador**);**

//printf("%.2f /%.2f ==%.2f\n",numerador,denominador,result);

**return** result**;**

**}**

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

float total **=** 0**;**

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

**{**

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

**}**

**return** total**;**

**}**

float getProbability**(**bitset**<**BIT\_IND**>** individual**,**int totalAptitude**){**

float aux**;**

aux**=(**float**)(**getIndividualAptitude**(**individual**)/(**float**)**totalAptitude**);**

// printf("%.2f / %i =%.2f\n",getIndividualAptitude(individual),totalAptitude,aux);

**return** aux**;**

**}**

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**];**

**}**

void swapInd**(**bitset**<**BIT\_IND**>** set**[]){**

array**<**int**,** NUM\_IND**>** arr**;**

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

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

arr**[**i**]** **=** set**[**i**].**to\_ulong**();**

**}**

random\_shuffle**(**arr**.**begin**(),** arr**.**end**());**

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

set**[**i**]** **=** arr**[**i**];**

**}**

**}**

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

float p **=** **(**float**)(**rand**()%**NUM\_RAND**)** **/(**float**)**NUM\_RAND **;**

printf**(**"%.2f\n"**,**p**);**

**if((**getIndividualAptitude**(**p1**)** **<** getIndividualAptitude**(**p2**))** **&&** p **>=** 0.70**){**

**return** p2**;**

**}**

**else** **if** **((**getIndividualAptitude**(**p1**)** **<** getIndividualAptitude**(**p2**))** **&&** p **<** 0.70**){**

**return** p1**;**

**}**

**else** **if** **((**getIndividualAptitude**(**p1**)** **>** getIndividualAptitude**(**p2**))** **&&** p **>=** 0.70**){**

**return** p1**;**

**}**

**else{**

**return** p2**;**

**}**

**}**

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**);**

**}**