Logotipo

Descripción generada automáticamente con confianza mediaImagen que contiene Logotipo

Descripción generada automáticamenteINSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

(ESCOM)

INGENIERIA EN SISTEMAS COMPUTACIONALES

ALGORTIMOS Y ESTRUCTURAS DE DATOS

2CV3

EVALUACION PRIMER PARCIAL:

“BUBBLE SORT, SELECTION SORT, INSERTION SORT CON DATOS BRINDADOS POR UN ARCHIVO .TXT “

ALUMNO:

CORTES BUENDIA MARTIN FRANCISCO

PROFESOR: SANCHEZ GARCIA OCTAVIO

04/11/22

ÍNDICE

Introducción 3

Código comentado 4

Capturas de tiempo 9

Tabla de comparaciones de los ordenamientos 10

Conclusiones 10

INTRODUCCION

En este primer examen consideraremos la base de datos compartida, la cual contempla los datos de una muestra de edades de personas adultas en USA.

(en este caso la nombre examen.txt).

Realizaremos el ordenamiento de estas edades mediante los algoritmos de

selección, inserción y burbuja.

Dentro del código colocaremos una opción para medir el tiempo de procesamiento de cada uno de los

algoritmos a generar, desde el inicio del ordenamiento y hasta el final de este.

La base de datos contienen un total de 2000 edades de personas adultas de USA. La cual la agregaremos con “FILE” en nuestro codigo como en un momento pondre, esto para poder ocuparla en nuestro arreglo

Para contar el tiempo ocuparemos “clock\_t” para poder contar desde el inicio del algoritmo hasta el final

Ahora recordemos un poco lo que hace cada odenamiento:

El ordenamiento “Bubble sort”: funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada.

El ordenamiento “Selection sort”: Consiste en encontrar el menor de todos los elementos del vector e intercambiarlo con el que está en la primera posición. Luego el segundo más pequeño, y así sucesivamente hasta ordenarlo todo.

El ordenamiento “Insertion sort”: La idea de este algoritmo de ordenación consiste en ir insertando un elemento de la lista ó un arreglo en la parte ordenada de la misma, asumiendo que el primer elemento es la parte ordenada, el algoritmo ira comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.

CODIGO COMENTADO

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main() {

//declaramos variables enteras utilizada en el codigo

int a[2000];

int A[2000];

int numero;

int i=0;

int j=0;

int k= 0;

int n=0;

int aux=0;

int min=0;

int tam=0;

int eliminar =2;

int verificar;

clock\_t begin,end,inicio,fin,in,out; //declaramos las variables para medir el tiempo

double time\_bub,time\_inser,time\_sel; //declaramos variable que nos dira el tiempo que tardo

FILE\*ex= fopen("examen.txt","rt"); //usamos el archivo txt para usarlo como base de datos con rt por que vamos a leerlo

printf("Ingrese el ordenamiento que quiera utilizar para acomodar : \n\n");

printf("1) Por Bubble Sort \n");

printf("2) Por Selection Sort \n");

printf("3) Por Insertion Sort (Recuerde que en este muestra de menor a mayor y menor a mayor )\n");

scanf("%i",&numero);

switch(numero){

//inicie un switch para que fuera un menu de seleccion

case 1 :

begin = clock(); //inicia a contar

do{

verificar = fscanf(ex,"%d",a+n);

if (verificar>0){

n++;

}

}while(verificar>0); //en este do while, revisara el archivo txt dado para guardarlos en el arreglo

while(n>eliminar - 1) //inicio de bubble sort

{

for (i=0; i<=n-eliminar; i++)

{

if(a[i]>a[i+1])

{

aux= a[i];

a[i]=a[i+1];

a[i+1] = aux;

}

}

eliminar++;

}

end = clock(); //Fin de contar tiempo

printf("Su arreglo en bubble sort es: \n\n");

for (i=0; i<n;i++) {

printf("%d ,", a[i]); //muestra el arreglo

}

time\_bub = (double)(end-begin)/CLOCKS\_PER\_SEC; //aqui hara las operaciones para pasarlo a segundos

printf("\n");

printf("\n");

printf("el tiempo fue de %.8f seconds.", time\_bub); //muestra el tiempo

printf("\n");

printf("\n");

return 0;

break;

case 2 :

inicio = clock(); //inicia a contar

do{

verificar = fscanf(ex,"%d",a+n);

if (verificar>0){

n++;

}

}while(verificar>0); //en este do while, revisara el archivo txt dado para guardarlos en el arreglo

for (i=0; i<n; i++) //inicio de selection sort

{

for(j=i+1; j <n; j++)

{

if(a[i]>a[j]){

aux= a[i];

a[i]=a[j];

a[j]=aux;

}

}

}

fin = clock(); //termina de contar

printf("Su arreglo con Selection sort: \n\n");

for (i=0; i<n;i++) {

printf("%d, ", a[i]);

} //muestra arreglo

time\_sel = (double)(fin-inicio)/CLOCKS\_PER\_SEC; //aqui hara las operaciones para pasarlo a segundos

printf("\n");

printf("\n");

printf("el tiempo fue de %.8f seconds.", time\_sel); //muestra el tiempo

printf("\n");

return 0;

break;

case 3 :

in = clock(); //inicia a contar

do{

verificar = fscanf(ex,"%d",A+tam);

if (verificar>0){

tam++;

}

}while(verificar>0); //en este do while, revisara el archivo txt dado para guardarlos en el arreglo

for(j= 1; j < tam; j++) { //inicio de insertion sort

min = A[j];

for (i=j-1;i >= 0 && A[i]> min;i = i-1)

{

A[i+1]= A[i];

}

A[i+1] = min;

}

out = clock();//finaliza de contar

printf("\n");

printf("INSERTION SORT DE Menor A Mayor: \n\n");

for (i=0;i<tam; i++){

printf("%d ,",A[i]); //muestra arreglo de menor a mayor

}

printf("\n");

printf("\n");

time\_inser = (double)((out-in)/CLOCKS\_PER\_SEC); //aqui hara las operaciones para pasarlo a segundos

printf("el tiempo fue de %.8f seconds.", time\_inser); //muestra el tiempo

printf("\n");

printf("\n");

printf("INSERTION SORT DE Mayor a Menor: \n\n");

for (i=tam-1;i>=0; i--){

printf("%d, ",A[i]);

} //muestra arreglo de menor a mayor

time\_inser = (double)((out-in)/CLOCKS\_PER\_SEC); //aqui hara las operaciones para pasarlo a segundos

printf("\n");

printf("\n");

printf("el tiempo fue de %.8f seconds.", time\_inser);//muestra el tiempo

printf("\n");

break;

default: printf("\n\n Ese numero no es una opcion, ingrese otro \n\n");

}

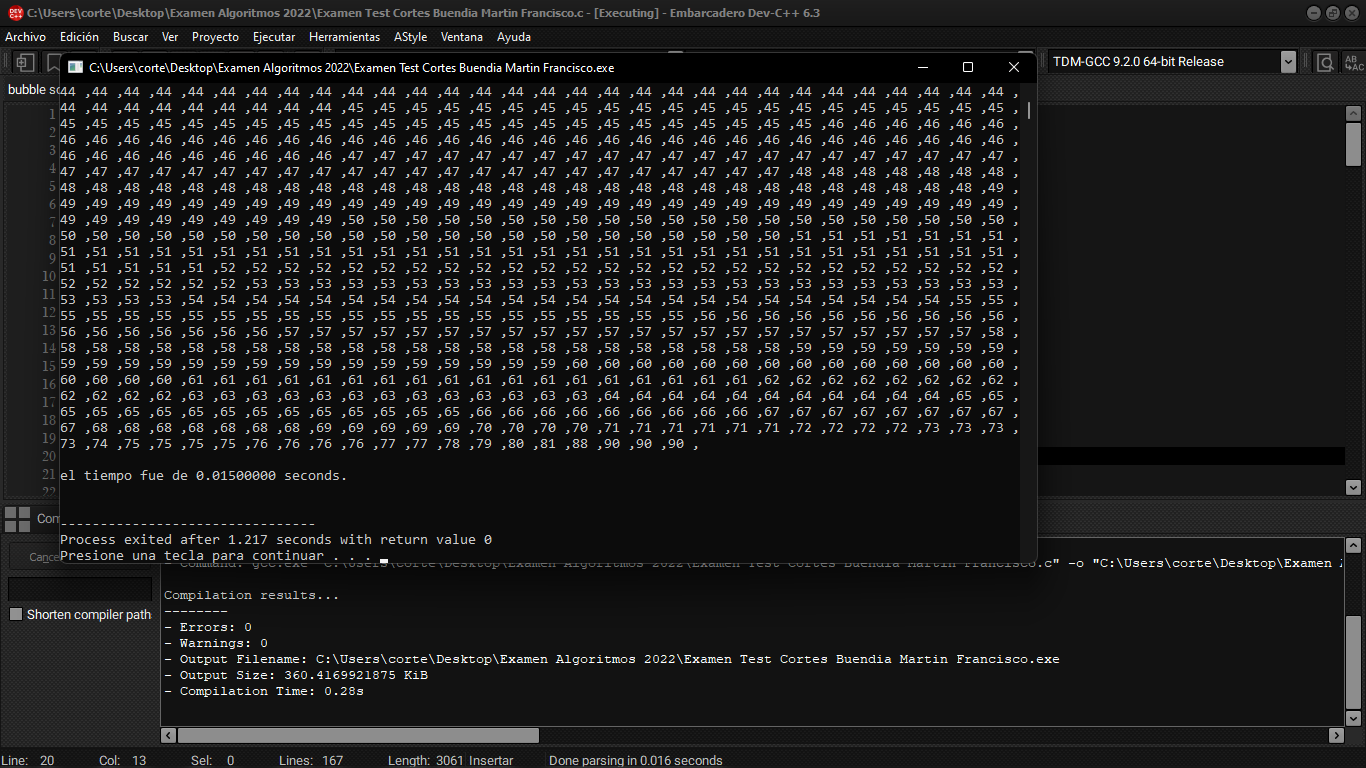
return 0;

}

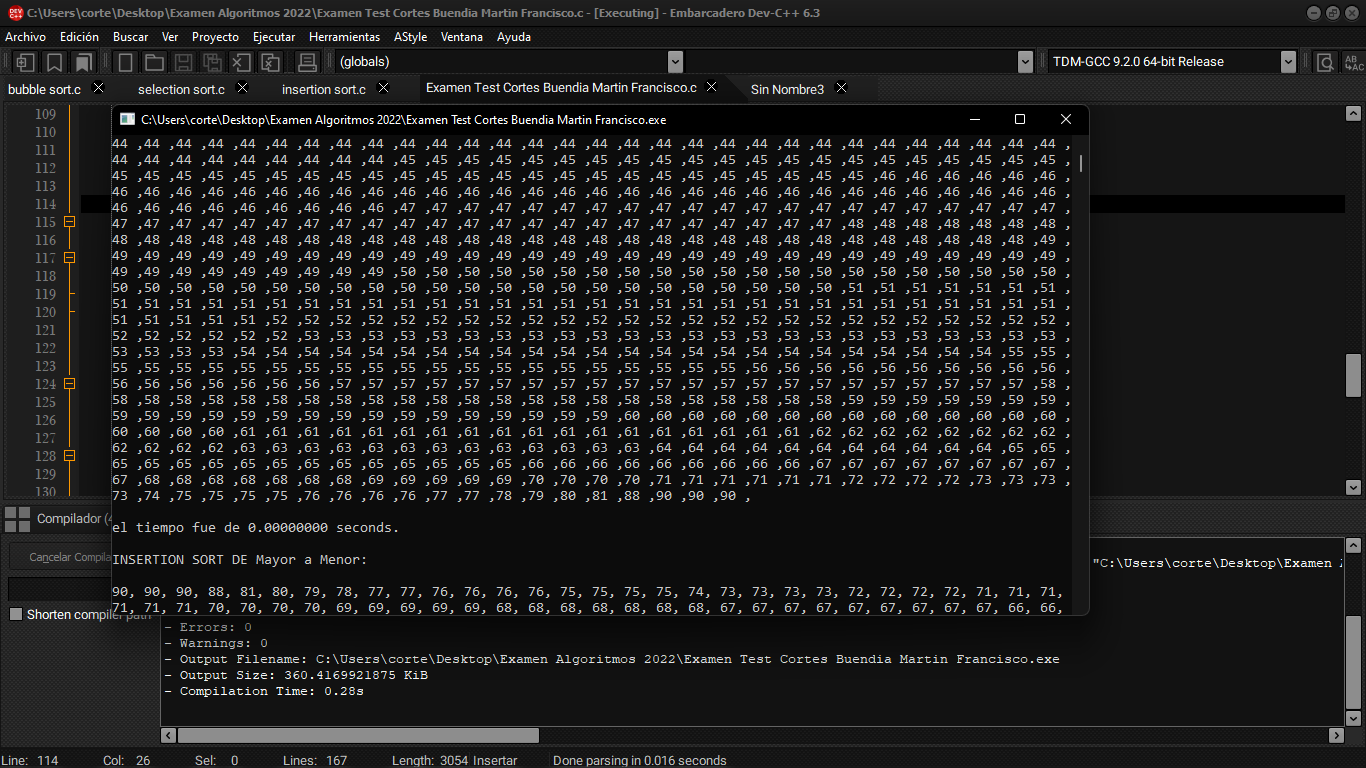
Captura de pantalla de computadora

Descripción generada automáticamente

***-Tiempo de Bubble sort***



***-Tiempo de Selection sort***



***-Tiempo de insertion sort***

TABLA DE COMPARACION DE ORDENAMIENTOS

|  |  |
| --- | --- |
| Algoritmo de ordenamiento | Tiempo |
| Bubble Sort | 1.6 csegundos |
| Selection Sort | 1.5 csegundos |
| Insertion Sort | 0 segundos |

Conclusiones

Como podemos notar el ordenamiento de Bubble sort tardo un poco mas ya que va analizando a todos y ordena si el primero es mayor al que le sigue, así hasta dar las vueltas necesarias para acabar y ordenar a todos. A diferencia del de inserción, el de inserción a cómo va avanzando va acomodando todo el arreglo, cuando el más grande llega al último lugar, ya ordeno todo el arreglo.