// ----------------------------------------------------------------------------

//

// Implementacion en C del algoritmo dar\_la\_vuelta

//

// ALGORITMIA: PRACTICA DE ANALISIS DE ALGORITMOS

//

// Copyright: Raquel Cortina

// Fecha: 1 de setiembre de 2019

//

// Programa para medir los tiempos de ejecucion de la implementacion en C

//

// ------------------------------------------------------------------------------------------

//

// La libreria time.h ofrece una funcion para medir tiempos. Su nombre es clock.

//

// clock\_t clock(void);

//

// Esta funcion devuelve el tiempo empleado por el procesador (en tics de reloj)

// hasta el punto en el que se efectua la llamada a la funcion clock. Para determinar

// el tiempo en segundos, el valor retornado por la funci�n clock debe ser dividido

// por el valor de la macro CLOCKS\_PER\_SEC.

//

// La libreria stdlib.h ofrece la funcion rand que selecciona un numero al azar. Si queremos

// un numero aleatorio entre un rango personalizado, por ejemplo entre 0 y 1000, utilizaremos

// el operador modulo (%) del siguiente modo:

//

// rand()%1000

//

// ------------------------------------------------------------------------------------------

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Fijamos el numero de tallas a medir

# define NUM\_TALLAS 12

// Fijamos el numero de repeticiones

# define REPETICIONES 100

// Prototipos de funciones

int rellena(int \*, int);

int dar\_la\_vuelta(int \*, int);

// Funcion principal

int main()

{

 int i,j;

 clock\_t tiempo\_inicial, tiempo\_final;

 // Se declara un vector de NUM\_TALLAS que son las que vamos a medir

 int talla[NUM\_TALLAS]={1000,2000,3000,4000,5000,6000,7000,8000,9000,10000, 11000,12000};

 // Mostramos la cabecera de la tabla en la que figuraran tallas vs. tiempos

 printf("\n\nDAR LA VUELTA A UN VECTOR\n\n");

 printf("Tiempo empleado:\n\n\n");

 printf("\t\tTalla\t\tTiempo\n\n");

 printf("\t\t-----\t\t------\n\n");

 // Bucle que recorre las diferentes tallas a medir

 for (i=0;i<NUM\_TALLAS;i++)

     {

     // Reservamos memoria para talla[i] enteros

     int \*V=(int \*) malloc (talla[i] \* sizeof(int));

     // Rellenado de un vector de talla[i]

     rellena(V,talla[i]);

     // Llamada a la funcion clock antes de la franja de codigo a medir

     tiempo\_inicial=clock();

     // Bucle para repetir el experimento

     for (j=1;j<=REPETICIONES;j++)

         // Invocacion de la funcion a medir

         dar\_la\_vuelta(V, talla[i]);

     // Llamada a la funcion clock despues de la franja de codigo a medir

     tiempo\_final=clock();

     // Liberamos el espacio reservado para el vector

     free(V);

     // Mostramos la talla medida y el tiempo empleado (tiempo medido/repeticiones)

     printf("\t\t%d\t\t%f\n", talla[i],(tiempo\_final-tiempo\_inicial) /(double)CLOCKS\_PER\_SEC / REPETICIONES);

     }

  return 0;

}

// Funcion que rellena el vector

int rellena(int \*V, int talla)

{

    int i;

    for (i=1;i<=talla;i++)

        V[i-1]=rand()%1000;

    return 0;

}

// Funcion que da la vuelta al vector

int dar\_la\_vuelta(int \*V,int talla)

{

    for (unsigned int i=0; i<talla-1; i++) {

        for (unsigned int j=0; j<talla-1; j++) {

            unsigned int aux = V[j];

            V[j] = V[j+1];

            V[j+1] = aux;

        }

    }

    return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

# define NUM\_TALLAS 10

# define REPEITERA 100000

# define REPERECUR 10000

# define REPEBUSCA 1000

// Prototipos de funciones

int   rellena(int \*\*, int);

int  SumaIter(int \*\*, int);

int SumaRecur(int \*\*, int);

bool    Busca(int \*\*, int, int);

// Programa principal

int main()

{

 int i,j;

 clock\_t tiempo\_ini1, tiempo\_ini2, tiempo\_fin1, tiempo\_fin2;

 int talla[NUM\_TALLAS]={1000,2000,3000,4000,5000,6000,7000,8000,9000,10000};

 int \*\*Matriz=NULL, sumaiter=0, sumarecur=0;

 printf("\n\n   SESION 2 DE PRACTICAS DE ANALISIS DE ALGORITMOS  \n\n");

 printf("Tiempo empleado:\n\n\n");

 // ALUMNO: COMENTAR/DESCOMENTAR LOS SIGUIENTES printf EN FUNCION DE LAS MEDIDAS A REALIZAR

 printf("\t\tTalla\t\tTiempo Mejor caso\tTiempo Peor caso\n");

 printf("\t\t-----\t\t----------------\t----------------\n");

/\*

 printf("\t\tTalla\t\tTiempo Iterativo\tTiempo Recursivo\n");

 printf("\t\t-----\t\t----------------\t----------------\n");

\*/

 for (int i=0;i<NUM\_TALLAS;i++) {

    // ALUMNO: RESERVA DINAMICA DE MEMORIA PARA UNA MATRIZ CUADRADA DE talla[i] x talla[i] ENTEROS

    Matriz = (int \*\*) malloc (talla[i] \* sizeof (int\*));

    for (j=0; j<talla[i]; j++) {

        Matriz[j] = (int \*) malloc (talla[i] \* sizeof (int));

    }

    rellena(Matriz,talla[i]);

    // FRAGMENTO DE CODIGO PARA ANALIZAR LA BUSQUEDA EN LA MATRIZ

    tiempo\_ini1=clock();

    for (j=1;j<=REPEBUSCA;j++)

      Busca(Matriz, talla[i], 4);       //Peor caso

    tiempo\_fin1=clock();

    tiempo\_ini2=clock();

    for (j=1;j<=REPEBUSCA;j++)

      Busca(Matriz, talla[i], 0);       //Mejor caso

    tiempo\_fin2=clock();

    printf("\t\t%d\t\t%f", talla[i],(tiempo\_fin1-tiempo\_ini1) /(double)CLOCKS\_PER\_SEC / REPEBUSCA);

    printf("\t\t%f\n",              (tiempo\_fin2-tiempo\_ini2) /(double)CLOCKS\_PER\_SEC / REPEBUSCA);

    // FRAGMENTO DE CODIGO PARA ANALIZAR LA SUMA DE LOS ELEMENTOS DE LA DIAGONAL DE LA MATRIZ

/\*

    tiempo\_ini1=clock();

    for (j=1;j<=REPEITERA;j++)

      // ALUMNO: LLAMADA A LA FUNCION SumaIter, ESTA FUNCION TIENE QUE DEVOLVER EL RESULTADO

      // PARA COMPARARLO CON EL DE LA VERSION RECURSIVA

      // sumaiter = SumaIter(...);

      sumaiter = SumaIter(Matriz,talla[i]);

    tiempo\_fin1=clock();

    tiempo\_ini2=clock();

    for (j=1;j<=REPERECUR;j++)

      // ALUMNO: LLAMADA A LA FUNCION SumaRecur, ESTA FUNCION TIENE QUE DEVOLVER EL RESULTADO

      // PARA COMPARARLO CON EL DE LA VERSION ITERATIVA

      // sumarecur = SumaRecur(...);

      sumarecur = SumaRecur(Matriz, talla[i]);

    tiempo\_fin2=clock();

    if (sumarecur != sumaiter) { printf("Inconsistencia en las sumas\n\n"); return -1; }

    printf("\t\t%d\t\t%f", talla[i],(tiempo\_fin1-tiempo\_ini1) /(double)CLOCKS\_PER\_SEC / REPEITERA);

    printf("\t\t%f\n",              (tiempo\_fin2-tiempo\_ini2) /(double)CLOCKS\_PER\_SEC / REPERECUR);

\*/

    //Matrix free

    for (int j=0; j<talla[i]; j++) {

        free (Matriz[j]);

    }

    free(Matriz);

 }

 return 0;

}

// Definiciones de funciones

int rellena(int \*\*M, int n){

    for (int i=0; i<n; i++) {

        for (int j=0; j<n; j++) {

            M[i][j]=0;

        }

    }

    return 0;

}

bool Busca(int \*\*M, int x, int n){

    int i = 0;

    int j = 0;

    bool encontrado = false;

    while (i < n) {

        while (j < n) {

            if (M[i][j] == x) {

                return true;

            }

            j=j+1;

        }

        i=i+1;

    }

    return false;

}

int SumaIter(int\*\*M, int n){

    int i=0;

    int suma=0;

    for (i=0;i<n;i++) {

        suma=suma+M[i][i];

    }

    return suma;

}

int SumaRecur(int \*\*M, int n){

    if (n==0) {

        return M[0][0];

    }

    return M[n-1][n-1] + SumaRecur(M,n-1);

}

**CModules:**

//Alloc Vector

int \*V=(int \*) malloc (talla[i] \* sizeof(int));

//Alloc Matrix

    Matriz = (int \*\*) malloc (talla[i] \* sizeof (int\*));

    for (j=0; j<talla[i]; j++) {

        Matriz[j] = (int \*) malloc (talla[i] \* sizeof (int));

    }

//Free Vector

free(V);

//Free Matrix

    for (int j=0; j<talla[i]; j++) {

        free (Matriz[j]);

    }

    free(Matriz);

// Definiciones de funciones

int rellena(int \*\*M, int n){

    for (int i=0; i<n; i++) {

        for (int j=0; j<n; j++) {

            M[i][j]=0;

        }

    }

    return 0;

}

//Print Matrix

int imprime (int \*\*M, int n){

    for (int i=0; i<n; i++) {

        for (int j=0; j<n; j++) {

            printf("%d ", M[i][j]);

        }

        printf("\n");

    }

    return 0;

}