Gonzalo Nieto 13/03/2024

[**main.c**](#_tdltukufcat0) **1**

[**Preguntas**](#_x14btb57t4xq) **4**

[**Ejemplos**](#_mvam02a53hpb) **5**

[**Juego de pruebas**](#_2w89g0dyecag) **5**

# **main.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#include <ctype.h> // Para utilizar la función isdigit

#define ALEAT 0

#define CREIX 1

#define DECREIX 2

/\*\*

\* @brief Genera una taula amb n enters

\* @param n (E) El nombre d'enters

\* @param taula (S) La taula d'enters

\* @return boolea cert -> s'ha creat la taula, fals -> altrament

\*/

bool genera(int \*\*taula, int n)

{

bool ok = true;

\*taula = malloc(n \* sizeof(int));

if (\*taula != NULL)

{

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

{

(\*taula)[i] = rand() % 10000;

}

}

else

ok = false;

return ok;

}

/\*\*

\* @brief Mostra per pantalla una taula amb n enters

\* @param taula (E) La taula d'enters

\* @param n (E) El nombre d'enters

\*/

void mostra(int taula[], int n)

{

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

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

printf("\n");

}

/\*\*

\* @brief Ordena una taula amb n enters de forma creixent amb el mètode selecció

\* @param taula (E) La taula d'enters

\* @param n (E) El nombre d'enters

\* @return int El nombre d'iteracions que fa

\*/

int ordena(int taula[], int n)

{

long iteracions = 0;

for (int i = 0; i < n - 1; i++)

{

int min\_j = i;

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

{

iteracions++; // Incrementamos el número de iteraciones

if (taula[j] < taula[min\_j])

min\_j = j;

}

int temp = taula[i];

taula[i] = taula[min\_j];

taula[min\_j] = temp;

}

return iteracions;

}

int main(int argc, char \*argv[])

{

int \*dades;

int mida = 0;

bool tot\_ok;

double tg\_ini, tg\_fi; // Per desar les mesures de temps de generar

double to\_ini, to\_fi; // Per desar les mesures de temps d'ordenar

long iteracions; // Per desar el nombre d'iteracions

if (argc == 2)

{

// Validar si el argumento es un número entero

for (int i = 0; argv[1][i] != '\0'; i++) {

if (!isdigit(argv[1][i])) {

printf("El argumento debe ser un número entero.\n");

return 1; // Salir del programa si no es un número entero

}

}

mida = atoi(argv[1]);

// COMPLETAR: afegiu aquí la mesura de temps tg\_ini

tg\_ini = clock(); // Medimos el tiempo de inicio de generación de datos

tot\_ok = genera(&dades, mida);

tg\_fi = clock(); // Medimos el tiempo de finalización de generación de datos

if (tot\_ok)

{

// COMPLETAR: afegiu aquí la mesura de temps tg\_fi

to\_ini = clock(); // Medimos el tiempo de inicio de ordenación

iteracions = ordena(dades, mida);

to\_fi = clock(); // Medimos el tiempo de finalización de ordenación

// Descomenteu per comprovar que ordenar( ) funciona amb valors de n petits

printf("Dades originals\n");

mostra(dades, mida);

// COMPLETAR: afegiu aquí la mesura de temps to\_ini

iteracions = ordena(dades, mida);

// COMPLETAR: afegiu aquí la mesura de temps to\_fi

// Descomenteu per comprovar que ordenar( ) funciona amb valors de n petits

//printf("Dades ordenades\n");

//mostra(dades, mida);

printf("\tTemps de generar %f\n", (tg\_fi - tg\_ini) / CLOCKS\_PER\_SEC);

printf("\tTemps d'ordenar %f\n", (to\_fi - to\_ini) / CLOCKS\_PER\_SEC);

printf("\tIteracions d'ordenar %ld\n", iteracions);

free(dades); // Alliberem la memòria

}

else

{

printf("No s'ha pogut reservar l'espai per a les dades.\n");

}

}

return 0;

}

# **Preguntas**

1. En funció del nombre d’elements de la taula d’enters, què consumeix

més temps: emplenar la taula o ordenar la taula? Quin és el motiu?

En general, llenar una tabla con números aleatorios suele ser más rápido que ordenarla. Esto se debe a que la generación de números aleatorios generalmente involucra operaciones simples y lineales, mientras que el proceso de ordenación implica iteraciones y comparaciones adicionales que pueden aumentar el tiempo de ejecución, especialmente para tablas grandes.

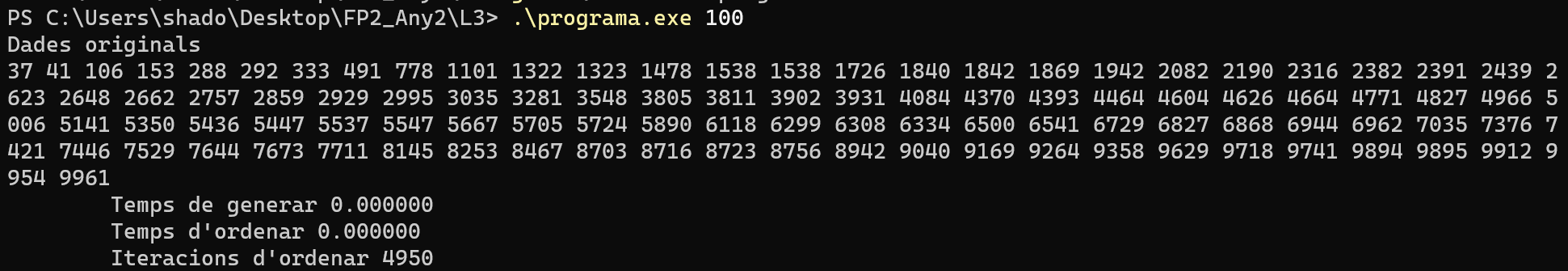
2. Quantes iteracions fa ordenar( ) en funció dels diferents valors per a

nombre d’elements que heu testejat?

Esto va a depender del tamaño de la tabla. En esta práctica utilizamos el algoritmo de selección, que realiza aproximadamente n^2/2 iteraciones, donde n es el número de elementos en la tabla. Por lo tanto, la cantidad de iteraciones aumenta cuadráticamente con el tamaño de la tabla.

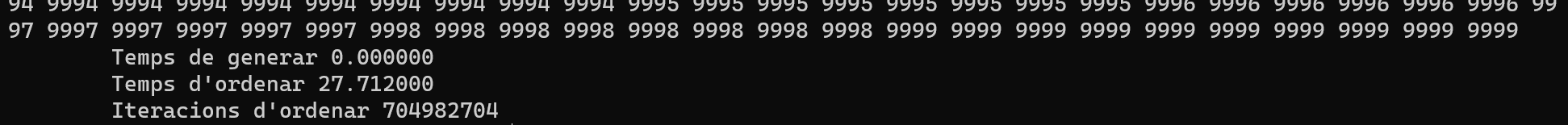
# **Ejemplos**

Con valor 100:



Con valor 100000:





# **Juego de pruebas**

| Caso de Prueba | Entrada | Salida Esperada | OK? |
| --- | --- | --- | --- |
| 1 | ./nombre\_programa 10.5 | "El argumento debe ser un número entero." | OK |
| 2 | ./nombre\_programa abc | "El argumento debe ser un número entero." | OK |
| 3 | ./nombre\_programa 100 | (Salida del programa con las mediciones de tiempo y el número de iteraciones) | OK |
| 4 | ./nombre\_programa 5000 | (Salida del programa con las mediciones de tiempo y el número de iteraciones) | OK |
| 5 | ./nombre\_programa 10000 | (Salida del programa con las mediciones de tiempo y el número de iteraciones) | OK |