Contenido

1. Objetivo de la modificación

El objetivo de la modificación es **modificar la práctica 2** de forma que permita un nuevo método de ordenación que corresponde al denominado método de ordenación Rápida (QuickSort).

2. Descripción del problema

El estudio de costes de tiempos se va a realizar (al igual que en las prácticas realizadas a lo largo del curso) sobre arrays de enteros generados aleatoriamente.

Para realizar la modificación tendréis que implementar un nuevo método llamado **ordenaQuickSort** que implemente el método indicado.

La implementación correspondiente debe seguir el pseudocódigo especificado a continuación (sacado de los apuntes de teoría).

3. Algoritmo para la ordenación Rápida (ordenaQuickSort)

El algoritmo de ordenación Rápida (**ordenaQuickSort**), para ordenar un vector a[1..n] de enteros, se basa en la estrategia “divide y vencerás”. Tomamos un valor de *x* (llamado pivote) y separamos los valores que son mayores o iguales que *x* a la derecha y los menores a la izquierda. Está claro que ahora solo hace falta ordenar los elementos en cada uno de los subrangos.

La elección del elemento pivote se puede seleccionar de diferentes formas (el mayor de los 2 primeros elementos distintos encontrados, el primer elemento, el último, el elemento medio, de forma aleatoria, mediana de 3 elementos, etc.).

Para el **caso medio**, el orden de complejidad del algoritmo es de **O(n log n).**

- El procedimiento ordenación Rápida (QuickSort), puede ser implementado como sigue:

Quicksort: algoritmo.

```
QuickSort(A,p,r)
   if (p < r) {
        q = Partition(A, p, r)
        QuickSort(A, p, q)
        QuickSort(A, q+1, r)
        }
```

Llamada inicial:

QuickSort(A, 0, n-1);

n: número de elementos del vector A

```
Partition(A, p, r)
  piv = A[p]; i = p − 1; j = r + 1;
  do{
    do{
      j = j − 1;
    }while (A[j] > piv)
    do{
      i = i + 1;
    }while (A[i] < piv)
    if (i < j) {
      A[i] ⟷ A[j]; //intercambiar valores
    }
  }while (i < j)
  return j;
```

4. Pasos a seguir para la modificación.

1. Crear un proyecto nuevo vacío.
2. Tomar los fuentes de vuestra práctica 2.
3. Implementar el método **`ordenaQuickSort`** según el pseudocódigo del apartado anterior.
4. Modificar el programa principal incorporando los menús en el apartado correspondiente.
5. Una vez completado el código, crear en el directorio ***Apellido1Apellido2*** dos carpetas:
   a. **fuentes**: con los .h y .cpp.
   b. **datos**: con el ejecutable y los datos.
6. Subir ***Apellido1Apellido2.rar*** en la tarea puesta al efecto.

5. Pautas para la evaluación del examen

Para aprobar el examen se requiere que:

P1.1.(40%) El programa se pueda ***compilar sin errores***.

P1.2.(60%) El programa debe ***funcionar correctamente***, sin colgarse y producir ***resultados correctos*** para el conjunto de pruebas a realizar

6. ANEXO DE AYUDA

RESUMEN ENUNCIADO DEL EXAMEN GRUPO L1:

Modificar la práctica 2 de forma que permita estudiar y comparar los siguientes algoritmos de ordenación:

-Burbuja
-Inserción
-Selección
-Ordenación Rápida (QuickSort) (nuevo)

Haz las modificaciones necesarias para que en los menús correspondientes aparezcan estos 4 métodos.

FICHEROS DE AYUDA (en amarillo marcamos lo nuevo a añadir)

AlgoritmosOrdenacion.h

```
#ifndef _ORDENACION
#define _ORDENACION

class AlgoritmosOrdenacion
{
public:
    AlgoritmosOrdenacion();
    ~AlgoritmosOrdenacion();

    void ordenaBurbuja(int v[], int size);
    void ordenaInsercion(int v[], int size);
    void ordenaSeleccion(int v[], int size);

    void ordenaQuickSort(int v[], int size);
    void QuickSort(int v[], int p, int r);
    int Partition(int v[], int p, int r);
};

#endif
```

AlgoritmosOrdenacion.cpp

```
#include "AlgoritmosOrdenacion.h"

AlgoritmosOrdenacion :: AlgoritmosOrdenacion() {}
AlgoritmosOrdenacion :: ~AlgoritmosOrdenacion(){}

...

void AlgoritmosOrdenacion :: ordenaQuickSort(int v[], int size) {
    QuickSort(v, 0, size-1);
}

void AlgoritmosOrdenacion :: QuickSort(int v[], int p, int r) {
  int q;
  if (p<r) {
    //A RELLENAR POR EL ALUMNO
  }
}

int AlgoritmosOrdenacion :: Partition(int v[], int p, int r) {
  //A RELLENAR POR EL ALUMNO
}
```