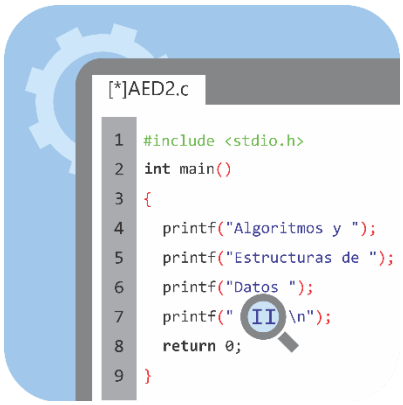

Algoritmos y Estructuras de Datos II

Trabajo Práctico 6. Ordenamiento.

OBJETIVOS:

- Conocer los conceptos de clasificación(ordenamiento).
- Aprender a analizar la complejidad de los distintos métodos.
- Aprender a implementar soluciones para el ordenamiento de arreglos tipificado de acuerdo con su categorización.

COMPETENCIAS

- Identificar, formular y resolver problemas mediante programación.
- Utilizar de manera efectiva técnicas y herramientas de aplicación para desarrollar software.
- Desempeñarse de manera efectiva en equipos de trabajo.
- Aprender en forma continua, autónoma y de manera colaborativa.

METODOLOGÍA

- El alumno deberá resolver individualmente los ejercicios propuestos.
- El alumno deberá codificar las soluciones en el lenguaje de programación C.
- Realizar consultas a través del canal de *slack* correspondiente a su comisión ó del aula virtual de la asignatura.

DURACIÓN

De acuerdo con la planificación de la asignatura, se deberá utilizar para la resolución de los ejercicios de esta serie, una clase práctica.

Ejercicios Propuestos

1. Escribir un programa que permita ingresar 10 valores reales por teclado. Luego escribir funciones que permitan:
   a) Ordenar un vector de menor a mayor por el método directo de burbuja.
   b) Ordenar un vector de menor a mayor por el método directo de selección.
   c) Ordenar un vector de menor a mayor por el método directo de inserción.

2. Desarrollar un programa que permita generar un arreglo de 15.000 números enteros aleatorios. Luego:
   a) Escribir una función para mostrar los elementos del array.

b) Escribir una función que reciba el array como parámetro y que permita ordenar los elementos de menor a mayor por el método directo de **burbuja**.
c) Escribir una función que reciba el array como parámetro y que permita ordenar los elementos de menor a mayor por el método directo de **selección**.
d) Escribir una función que reciba el array como parámetro y que permita ordenar los elementos de menor a mayor por el método directo de **inserción**.
e) Calcular el tiempo de ejecución de cada método, y mostrar por pantalla la duración en cada caso.

3. El código de la siguiente función está incompleto. Completar el código para que la función implemente el algoritmo de intercambio directo o burbuja.

```
void intercambioDirecto( tVectorFloat pVector ) {
    int i, j;
    float aux;

    for( i=0; i<MAX-1; i++ ) {
        for( j=0; j<MAX-1; j++ ) {
            if( pVector[j] > [          ] ) {
                aux = [          ];
                pVector[ [ ] ] =  pVector[j+1];
                pVector[ [ ] ] = [          ];
            }
        }
    }
}
```

4. El código de la siguiente función está incompleto. Completar el código para que la función implemente el algoritmo de ordenamiento de selección directa u obtención sucesiva de menores.

```
void seleccionDirecta( tVectorFloat pVector ) {
    int i, j, posMenor;
    float valorMenor, aux;
    for ( i=0; i<MAX-1; i++ ) {
        valorMenor = [          ];
        posMenor = [          ];
        for( j=i+1; j<MAX; j++) {
            if( pVector[j] [ ] valorMenor ) {
                valorMenor = [          ];
                posMenor = [          ];
            }
        }
        if( posMenor != i ) {
            aux = [          ];
            pVector[ [     ] ] = pVector[posMenor];
            pVector[ [     ] ] = aux;
        }
    }
}
```

5. El código de la siguiente función está incompleto. Completar el código para que la función implemente el algoritmo de búsqueda secuencial ordenada.

```
bool busquedaSecuencialOrdenada( tVectorFloat pVector, float pElem )
{
    int i = 0;
    bool existe = [          ];

    while( (pVector[i] < pElem) && [          ] ) {
        [          ];
    }
    if( pVector[i] == pElem ){
        existe = [          ];
    }
    return [          ];
}
```