

Programación I

Práctica 3

Curso 2020/21

Enunciado general

En esta práctica se implementará una aplicación que llamaremos “Integración de un sensor en una red de sensores” que permitirá calcular y gestionar la situación de un sensor con respecto a los cinco sensores más próximos en una WSN (Wireless Sensor Network) y la potencia con que ha de transmitir en función de la distancia a los sensores vecinos.

El programa se iniciará solicitando las distancias a los cinco sensores vecinos y escribiendo de nuevo las distancias introducidas para comprobar que se han leído correctamente. A continuación, se lanzará un menú con las siguientes opciones:

1. Calcular la potencia de transmisión
2. Modificar la distancia a un sensor vecino
3. Calcular el rango de la potencia de transmisión
4. Salir

Una vez ejecutas las acciones correspondientes a una opción del menú, se presentará de nuevo el menú hasta que se seleccione la opción 4, en cuyo caso el programa terminará.

Enunciado detallado y proceso de realización de la práctica

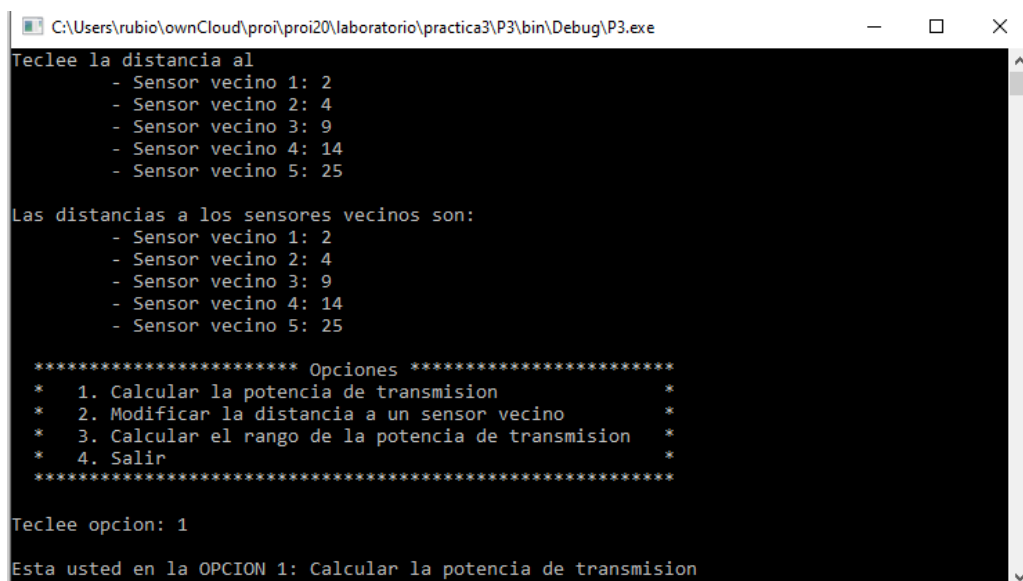
Para facilitar la realización de la práctica se propone su realización en 3 fases.

Fase 1

Se codificará el programa principal para que realice las siguientes acciones:

- Leer y almacenar las distancias de hasta los cinco sensores vecinos, teniendo en cuenta que son de tipo real y que sólo pueden estar comprendidas entre 0.5 y 30.0 metros.
- Escribir por la salida estándar las distancias leídas previamente, indicando el número de sensor vecino al que pertenece, teniendo en cuenta que la primera distancia corresponde al sensor 1, la segunda al 2 y así sucesivamente.
- Escribir el menú que se ha indicado en el aparatado anterior de forma que, si se elige una opción, salga por pantalla una frase indicando que el programa va a ejecutar esa opción y que vuelva a presentar el menú hasta que se elija la opción 4.

Por ejemplo:



```
C:\Users\rubio\ownCloud\proi\proi20\laboratorio\practica3\P3\bin\Debug\P3.exe
Teclee la distancia al
- Sensor vecino 1: 2
- Sensor vecino 2: 4
- Sensor vecino 3: 9
- Sensor vecino 4: 14
- Sensor vecino 5: 25

Las distancias a los sensores vecinos son:
- Sensor vecino 1: 2
- Sensor vecino 2: 4
- Sensor vecino 3: 9
- Sensor vecino 4: 14
- Sensor vecino 5: 25

***** Opciones *****
* 1. Calcular la potencia de transmision *
* 2. Modificar la distancia a un sensor vecino *
* 3. Calcular el rango de la potencia de transmision *
* 4. Salir *
*****

Teclee opcion: 1

Esta usted en la OPCION 1: Calcular la potencia de transmision
```

Fase 2

Codificar las funciones que se indican a continuación correspondientes a la lectura y escritura de las distancias e invocarlas desde el programa principal, sustituyendo la parte de código relacionada con estas acciones, codificada en la fase 1.

```
/*
 * Función: leerDistanciaSensoresVecinos
 * Lee de la entrada estándar las distancias a los 5 sensores vecinos
 * del sensor que se está configurando. Debe indicar el número del
 * sensor vecino (del 1 al 5)
 * Parámetros de entrada: ninguno
 * Parámetros de salida pasados por referencia:
 * distancias: array de reales con las 5 distancias.
 * Valor devuelto por la función: ninguno
 */
void leerDistanciaSensoresVecinos (double distancias[]);
```

```

/*
 * Función: escribirDistanciaSensoresVecinos
 * Escribe en la salida estándar las distancias a los 5 sensores vecinos
 * del sensor que se está configurando. Debe indicar el número del
 * sensor vecino (del 1 al 5)
 * Parámetros de entrada:
 * distancias: array de reales con las 5 distancias
 * Precondiciones: distancias tiene que estar inicializado
 * Parámetros de salida pasados por referencia: ninguno
 * Valor devuelto por la función: ninguno
 */
void escribirDistanciaSensoresVecinos (const double distancias[]);

```

Como ejemplo se proporciona el código de la función (que se encuentra en el P3Esqueleto.c que se proporciona en el espacio Moodle de la asignatura):

```

/*
 * Función: leerRealEnRango
 * Lee de la entrada estándar un número real en el rango [rangoInf, rangoSup].
 * Si se teclea un número fuera de rango lo indica y vuelve a solicitar un nuevo valor.
 * Parámetros de entrada:
 * rangoInf: real, rango inferior del número a leer
 * rangoSup: real, rango superior del número a leer
 * Precondiciones: ninguna
 * Parámetros de salida pasados por referencia: ninguno
 * Valor devuelto por la función: número real leído por la entrada estándar (teclado)
 */
double leerRealEnRango (double rangoInf, double rangoSup);

```

Se pide: *Codificar las acciones correspondientes a la opción 1 del menú, **Calcular la potencia de transmisión**.*

Para ello, además, se deberá codificar e invocar la función *calcularPotenciaTransmision* que se describe a continuación, y una vez obtenido el resultado de la potencia, este se debe escribir por la salida estándar.

```

/*
 * Función: calcularPotenciaTransmision
 * Calcula la potencia de transmisión del sensor teniendo en cuenta
 * que tiene que emitir con una potencia de 1.5μW por cada metro
 * de distancia. Por tanto, habrá que detectar cuál es el sensor vecino
 * que está a mayor distancia.
 * Parámetros de entrada:
 * distancias: array de reales con las 5 distancias
 * Precondiciones: distancias tiene que estar inicializado
 * Parámetros de salida pasados por referencia: ninguno
 * Valor devuelto por la función: valor real con la potencia de transmisión necesaria
 */
double calcularPotenciaTransmision (const double distancias[]);

```

Fase 3

Se pide: *Codificar las acciones correspondientes a la opción 2 del menú, **Modificar la distancia a un sensor vecino**, que se utilizará en caso de que sea necesario modificar la distancia a un sensor vecino porque esta hay modificado su posición.*

Para implementar esta funcionalidad se codificará la función *modificarDistanciaSensorVecino* con la siguiente funcionalidad e interfaz:

```
/*
 * Función: modificarDistanciaSensorVecino
 *      Pregunta si se desea modificar alguna distancia y en caso afirmativo se consulta el
 *      número del sensor del que se desea modificar y la nueva distancia, y se realiza la modificación
 * Parámetros de entrada/salida:
 *      distancias: array de reales con las 5 distancias
 *      Precondiciones: distancias tiene que estar inicializado
 * Parámetros de salida pasados por referencia:
 *      sensorVecino: número del sensor vecino cuya distancia se ha modificado (de 1 a 5)
 * Valor devuelto por la función: valor booleano, true si se ha modificado alguna distancia,
 *      false si no se ha hecho ninguna modificación
 */
bool modificarDistanciaSensorVecino ( double distancias[], int *sensorVecino);
```

En el cuerpo de acciones de esta función se debe leer el número del sensor vecino cuya distancia se desea modificar. Los números de los sensores vecinos son enteros del 1 al 5. Para realizar la lectura del número del parcial se invocará a la función *leerEnteroEnRango* con la siguiente funcionalidad e interfaz:

```
/*
 * Función: leerEnteroEnRango
 *      Lee de la entrada estándar un número entero en el rango [rangolnf, rangoSup].
 *      Si se teclea un número fuera de rango lo indica y vuelve a solicitar un nuevo valor.
 * Parámetros de entrada:
 *      rangolnf: Mínimo valor entero aceptado como válido
 *      rangoSup: Máximo valor entero aceptado como válido
 *      Precondiciones: Ninguna
 * Parámetros de salida pasados por referencia: ninguno
 * Valor devuelto por la función: Valor entero leído por la entrada estándar (teclado)
 */
/* El alumno determinará el prototipo y realizará la codificación*/
```

Fase 4

Se pide: *Codificar las acciones correspondientes a la opción 3 del menú, **Calcular el rango de la potencia de transmisión**, que expresará mediante un carácter el rango de potencia al que pertenece el sensor, según la siguiente tabla:*

Rango de potencia transmitida (μW)	Valor Alfabético	Significado
Menor que 10	L	Baja potencia
Desde 10 (incl.) hasta 20 (excl..)	M	Media potencia
Desde 20 (incl.) hasta 30 (excl..)	H	Alta potencia
30 ó más	V	Muy alta potencia

Para almacenar la potencia transmitida numérica de forma conjunta con su valor alfabético se definirá la estructura tRangoPotencia:

```
typedef struct {
    double valorNumerico;
    char valorAlfabetico;
} tRangoPotencia;
```

Para atender la funcionalidad de la opción **3** del menú se codificarán las funciones:

- a) **calcularRangoPotencia** que asignará el valor alfabético según el rango de potencia transmitida, que se expresa en la tabla anterior. Su funcionalidad e interfaz es:

```
/*
 * Función: calcularRangoPotencia
 *      Calcula el valor alfabético correspondiente a una potencia transmitida
 *      por un sensor y almacena ambas en una estructura del tipo tRangoPotencia
 * Parámetros de entrada:
 *      potenciaTransmitida: real que contiene la potencia transmitida
 *      Precondiciones: potenciaTransmitida debe estar calculada previamente
 * Parámetros de salida pasados por referencia: rangoPotencia de tipo tRangoPotencia
 * Valor devuelto por la función: ninguno
 */
void calcularRangoPotencia (double potenciaTransmitida, tRangoPotencia *rangoPotencia);
```

- b) **escribirRangoPotencia** que presentará por la salida estándar (pantalla) el valor numérico acompañado por el valor alfabético. Se deberá definir el prototipo de la función atendiendo a la descripción de la interfaz que figura a continuación:

```
/*
 * Función: escribirRangoPotencia
 *      Presenta por la salida estándar el valor numérico y el valor alfabético de modo conjunto
 * Parámetros de entrada:
 *      rangoPotencia: estructura de tipo tRangoPotencia
 *      Precondiciones: rangoPotencia debe de estar inicializado
 * Parámetros de salida pasados por referencia: ninguno
 * Valor devuelto por la función: ninguno
 */
void escribirRangoPotencia (tRangoPotencia rangoPotencia);
```

