

**UNIVERSIDAD POLITÉCNICA DE MADRID**

**ESCUELA TÉCNICA SUPERIOR DE**

**INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN**

Programación I

**Práctica 3**

**Semestre de otoño**

**Curso 2019/20**

**INDICE**

***Contenido***

Introducción 3

Objetivos 3

Duración 3

Requisitos 3

Enunciado general 3

Enunciado detallado y proceso de realización de la práctica 4

Fase 1 4

Fase 2 5

Fase 3 6

Fase 4 7

Consideraciones generales para la entrega y evaluación de la práctica 8

Notas sobre organización de espacios de trabajo, proyectos y archivos fuente 8

Realización de la práctica 8

Procedimiento de entrega y evaluación de la práctica 8

***Introducción***

***Objetivos***

Los objetivos de esta práctica son los siguientes:

* Iniciarse en el diseño de *algoritmos parametrizados*. Es especialmente importante cuidar la definición de los parámetros de entrada y/o salida.
* Asentar los conceptos que se han explicado en las clases de teoría sobre implementación de funciones (declaración y definición) en lenguaje C a partir de los algoritmos parametrizados. Incidir en la importancia de los comentarios en los prototipos de las funciones.
* Familiarizarse con el diseño e implementación de aplicaciones de complejidad media-baja, aplicando la metodología de *diseño descendente* a una aplicación que permite al usuario la interacción a través de un menú de opciones.
* Iniciarse en el manejo de *arrays* y estructuras.
* Continuar afianzando los conceptos de programación estudiados en teoría y/o prácticas anteriores, así como adquirir destreza con el manejo del IDE utilizado, *Code::Blocks*.

***Duración***

La duración de esta práctica es de tres sesiones presenciales (6 horas) más el tiempo de trabajo autónomo que el alumno precise para alcanzar los objetivos previstos. Se recomienda la consulta de la guía de la asignatura para tener una orientación sobre estos tiempos.

***Requisitos***

Haber realizado y entregado la práctica 2 y haber asistido a clase de teoría para manejar los conceptos tratados en esta práctica.

***Enunciado general***

En esta práctica se implementará una aplicación que llamaremos “Nota final de una asignatura” que permitirá calcular y gestionar la nota final de una asignatura a través de la nota obtenida en 4 exámenes parciales y la correspondiente ponderación de cada nota.

El programa se iniciará solicitando las notas de los 4 parciales y escribiendo de nuevo las notas introducidas para comprobar que se han leído correctamente. A continuación, se lanzará un menú con las siguientes opciones:

1. Calcular nota final
2. Modificar nota parcial
3. Calcular rango nota final
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. Al seleccionar la opción 4 el programa terminará.

***Enunciado detallado y proceso de realización de la práctica***

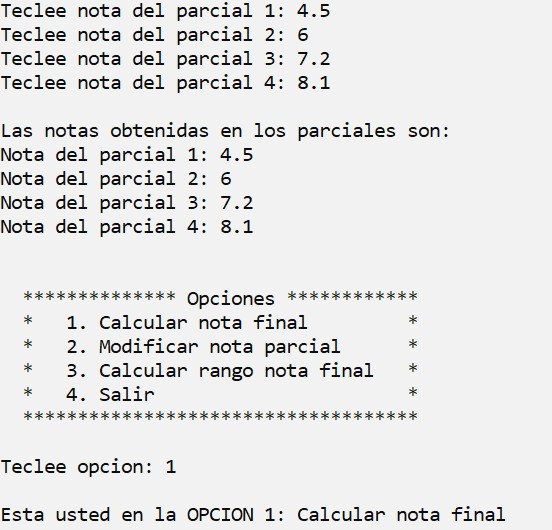
Para facilitar la realización de la práctica se propone su realización en 3 fases.

***Fase 1***

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

* Leer y almacenar las notas de 4 parciales, teniendo en cuenta que las notas son de tipo real y que sólo pueden estar comprendidas entre 0.0 y 10.0.
* Escribir por la salida estándar las notas leídas previamente, indicando el número del parcial al que corresponden.
* 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:



***Fase 2***

Codificar las funciones que se indican a continuación correspondientes a la lectura y escritura de las notas de los 4 parciales e invocarlas desde el programa principal, sustituyendo la parte de código relacionada con estas acciones, codificada en la fase 1.

/\*

* 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);

/\*

* Función: leerNotas Parciales
* Lee de la entrada estándar las notas de los 4 parciales de la asignatura,
* indicando el número de cada parcial (del 1 al 4)
* Parámetros de entrada: ninguno
* Parámetros de salida pasados por referencia:
* notas: array de reales con las notas de los 4 parciales
* Valor devuelto por la función: ninguno

\*/

void leerNotasParciales (double notas[]);

/\*

* Función: escribirNotasParciales
* Escribe en la salida estándar las notas de los 4 parciales de la asignatura,
* indicando el número de cada parcial (del 1 al 4)
* Parámetros de entrada:
* notas: array de reales con las notas de los 4 parciales
* Precondiciones: notas tiene que estar inicializado
* Parámetros de salida pasados por referencia: ninguno
* Valor devuelto por la función: ninguno

\*/

void escribirNotasParciales (const double notas[]);

Codificar las acciones correspondientes a la opción **1** del menú, **Calcular nota final**.

Para calcular la nota final será necesario declarar en main() una array constante con 4 valores de tipo entero que se correspondan con el porcentaje que pondera cada parcial. Estos valores son: 15, 15, 25 y 45.

Se deberá codificar e invocar la función que se describe a continuación, y una vez obtenido el resultado de la nota final, este se debe escribir por la salida estándar.

/\*

* Función: calcularNotaFinal
* Calcula la nota final de la asignatura haciendo la media ponderada
* de los 4 parciales (15%, 15%, 25% y 45%), respectivamente
* Parámetros de entrada:
* notas: array de reales con las notas de los 4 parciales
* ponderación: array de enteros con la ponderación de cada parcial
* Precondiciones: notas y ponderación tienen que estar inicializados
* Parámetros de salida pasados por referencia: ninguno
* Valor devuelto por la función: valor real con la nota final de la asignatura

\*/

double calcularNotaFinal (const double notas[], const int ponderacion[]);

***Fase 3***

En caso de que un alumno haya anotado mal una nota y deba corregirla, o que haya cambiado de nota a consecuencia de una revisión de exámenes, contamos con la opción 2 del menú, **Modificar notas parciales.**

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

/\*

* Función: modificarNotaParcial
* Pregunta si se desea modificar alguna nota y en caso afirmativo se consulta el
* número del parcial que se desea modificar y la nueva nota, y se realiza la modificación
* Parámetros de entrada:
* notas: array de reales con las notas de los 4 parciales
* Precondiciones: notas tiene que estar inicializado
* Parámetros de salida pasados por referencia:
* parcial: número del parcial cuya nota se ha modificado (de 1 a 4)
* Valor devuelto por la función: valor booleano, true si se ha modificado alguna nota,
* false si no se ha hecho ninguna modificación

\*/

bool modificarNotaParcial ( double notas[], int \*parcial);

En el cuerpo de acciones de esta función se debe leer el número del parcial cuya nota se desea modificar. Los números de los parciales son enteros del 1 al 4. 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 [rangoInf, rangoSup].
* Si se teclea un número fuera de rango lo indica y vuelve a solicitar un nuevo valor.
* Parámetros de entrada:
* rangoInf: 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)

\*/

***Fase 4***

Para atender la funcionalidad de la opción **3** del menú se codificará la función **calcularRangoNotaFinal** que asignará una nota alfabética según los siguientes rangos de notas:

|  |  |  |
| --- | --- | --- |
| Rango de nota | Nota Alfabética | Significado |
| Menor que 5 | I | Insuficiente (suspenso) |
| Desde 5 (incl.) hasta 7 (excl..) | A | Aprobado |
| Desde 7 (incl.) hasta 9 (excl..) | N | Notable |
| 9 o más | S | Sobresaliente |

Para almacenar la nota final numérica de forma conjunta con su nota alfabética se definirá la estructura tRangoNota: typedef struct {

double valorNota; char notaAlfabetica;

} tRangoNota;

La función calcularRangoNotafinal vendrá definida por la siguiente funcionalidad e interfaz:

/\*

* Función: calcularRangoNotaFinal
* Calcula la nota alfabética correspondiente a una nota final y almacena ambas en
* una estructura del tipo tRangoNota
* Parámetros de entrada:
* notaFinal: real que contiene la nota final
* Precondiciones: notaFinal debe estar calculada previamente
* Parámetros de salida pasados por referencia: rangoNota de tipo tRangoNota
* Valor devuelto por la función: ninguno

\*/

void calcularRangoNotaFinal (double notaFinal, tRangoNota \*rangoNota);

Por último, se codificará la función escribirRangoNotaFinal que presentará por la salida estándar (pantalla) la nota numérica acompañada por la nota alfabética. Se deberá definir el prototipo de la función atendiendo a la descripción de la interfaz que figura a continuación:

/\*

* Función: escribirRangoNotaFinal
* Presenta por la salida estándar la nota aritmética y la nota alfabética de modo conjunto
* Parámetros de entrada:
* rangoNota: estructura de tipo tRangoNota
* Precondiciones: rangoNota debe de estar inicializado
* Parámetros de salida pasados por referencia: ninguno
* Valor devuelto por la función: ninguno

\*/

***Consideraciones generales para la entrega y evaluación de la práctica***

***Notas sobre organización de espacios de trabajo, proyectos y archivos fuente***

De acuerdo con lo especificado en prácticas anteriores, y considerando que **es un único programa el que se realiza y entrega**:

1. Se creará **un único proyecto ‘P3’** dentro de la carpeta que cada alumno tenga para las prácticas de programación 1***.***
2. El archivo fuente se denominará ‘**p3.c’**.
3. La compilación, enlazado, ejecución y posible depuración de los programas se hará utilizando CodeBlocks.

***Realización de la práctica.***

La práctica **comienza el martes 22 de octubre** y se realizará en **grupos** de 2 alumnos (salvo excepciones que deben estar conformadas con el profesor del laboratorio).

***Procedimiento de entrega y evaluación de la práctica***

Los alumnos que forman un grupo deberán entregar **la misma versión de la práctica**.

La entrega se realizará en un único fichero de **extensión .c** que contendrá el archivo fuente generado. El nombre del fichero .c tendrá el siguiente formato, con el primer apellido de cada alumno y el grupo de laboratorio al que pertenece:

# p3\_Apellido1Apellido2\_grupo.c

**La práctica se debe entregar en la plataforma Moodle con fecha límite el jueves 21 de noviembre a las 10:00 AM.**

La evaluación de esta práctica se realizará en la prueba de evaluación continua del laboratorio del **25 de noviembre, en el aula de examen.**

# La no entrega de esta práctica supondrá un 0 en la mencionada prueba de evaluación continua del laboratorio.