Actividad: Laboratorio 2. Ejercicios avanzados con Dev C/C++

**Objetivos**

* Aprender a programar ejercicios manejando estructuras, arrays y matrices en el lenguaje de programación C.
* Utilizar el concepto de función para aplicar abstracción y modularidad en un programa.
* Aprender a resolver problemas matemáticos mediante algoritmos y llevarlos a la práctica mediante un programa en lenguaje C.

**Descripción de la actividad**

Preparación para el laboratorio:

Antes de acudir al laboratorio deberás haber instalado en tu máquina el compilador Dev-C++, haber leído el tutorial disponible en el apartado **Recursos externos** del tema sobre este compilador y haber ejecutado al menos el algoritmo propuesto como ejemplo correctamente. Además es aconsejable haber realizado el laboratorio #1.

**Nota**: aquellos alumnos que no dispongan del sistema operativo Windows pueden utilizar las siguientes herramientas:

* Usuarios de Mac OS X:
  + Xcode.
  + NetBeans.
* Usuarios de Linux:
  + NetBeans.

Estas herramientas están disponibles en el apartado **Recursos externos** del tema.

Descripción del laboratorio:

Una vez que ya puedan realizarse, compilarse y ejecutarse algoritmos en tu compilador, el laboratorio consiste en implementar la solución de algunos problemas a través de algoritmos en C. En este laboratorio deberás resolver ejercicios de vectores, matrices, funciones, punteros, ficheros y recursividad utilizando el compilador Dev-C++. Los ejercicios se conocerán en el momento de inicio de la sesión online del laboratorio.

**Entrega**

Una vez acabado el laboratorio, la entrega correcta del trabajo realizado consiste en la realización de un documento (utilizando la plantilla de Microsoft Word disponible en el aula virtual) explicando la realización de los diferentes ejercicios:

* Cómo se llega a la solución.
* Tipos de estructuras utilizadas.
* Justificación del uso de dichas estructuras.
* Comentarios sobre los posibles errores o problemas que hayas encontrado a la hora de la realización del laboratorio.

Además se adjuntará el código fuente final de la solución de cada uno de los problemas y el ejecutable final generado, donde aparecerá en pantalla el nombre del autor.

En todos los archivos debe de quedar claramente definido qué es cada archivo. Por ejemplo, el ejecutable del problema número 4 se llamará: *Saenz\_Gomez\_Juan\_Problema\_4*. Y una vez finalizado el laboratorio todos los archivos deberán ser incluidos en un archivo comprimido (.zip) con el nombre de la persona y el número de laboratorio correspondiente, por ejemplo: *Saenz\_Gomez\_Juan\_laboratorio\_2.zip*.

**Rúbrica**

|  |  |  |  |
| --- | --- | --- | --- |
| Ejercicios avanzados con Dev C/C++ (valor real: 4 puntos) | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Criterio 1 | El código compila | 4 | 40% |
| Criterio 2 | La estructura y diseño es adecuada | 2 | 20% |
| Criterio 3 | Los resultados devueltos son correctos | 3 | 30% |
| Criterio 4 | El programa es óptimo y está correctamente comentado | 1 | 10% |
|  |  | **10** | **100 %** |

**Extensión** **máxima:** la memoria deberá tener un máximo de 15 páginas, fuente Calibri 12 e interlineado 1,5.

**MEMORIA LABORATORIO 2**

**Ejercicio 1**

Problema: Implementar un programa que ofrezca un menú al usuario para realizar las siguientes operaciones con matrices

de números reales de tamaño 3x3:

1.Suma de dos matrices,A y B

2.Resta de dos matrices, A y B

3.Multiplicación de dos matrices, A y B

Resolución del problema: al tratarse de una matriz cuadrada, con filas y columnas definidas inicialmente por el problema, ambas variables podemos manejarlas como una constante con la directiva #define. Utilizaremos estas variables para componer nuestra matriz o arrays de dos dimensiones: matrix[][]

Para este ejercicio vamos a utilizar 5 funciones diferentes, las dos primeras nos servirán para interactuar con el usuario, y las otras 3 para operar sobre las matrices declaradas.

En todas las funciones tendremos que iterar sobre los campos de los arrays con bucles “for” anidados, para manejarnos indistintamente por el array de filas o el de columnas, y utilizaremos una función de salida llamada showMatrix() para mostrar el resultado al usuario.

Finalmente reutilizaremos el código del primer laboratorio para manejar la selección del usuario.

Comentarios: El principal problema que me he encontrado ha sido con el manejo de la matriz en las funciones declaradas, iterar y guardar cada operación, especialmente en el producto de matrices, me ha requerido investigar sobre ello y entender el código que habías compartido en la página web durante el laboratorio (y clase).

**Ejercicio 2**

Problema: Implementar un programa en C que defina las siguientes estructuras Domicilio y Persona, solicitando al usuario la información completa de una persona e imprimiendola en pantalla.

Resolución del problema: El paso inicial es definir las estructuras de datos que va a utilizar el programa de tal manera, que todos los campos estén correctamente definidos y tipados, mientras anidamos una estructura dentro de la otra. Después definimos la función que va a mostrar por pantalla los datos personales introducidos, llamada showPerson(), y declararemos una instancia p para abreviar y facilitar el uso de las variables introducidas en la petición de datos que vamos a realizarle al usuario con dos funciones diferentes; scanf() para aquellos campos que no requieran manejo de espacios en blanco, y fgets() para aquellos que puedan necesitar manejar un espacio en blanco entre los datos introducidos por el cliente (como en nombre, donde un usuario puede llamarse Juan Pedro por ejemplo).

Comentarios: Sinceramente ha sido el ejercicio que más dolores de cabeza me ha dado de los 4, en principio debería de haber sido de

los más sencillos, pero el manejo de los posibles errores (espacios en blanco), y el entendimiento de las diferencias entre estructuras con punteros y acceso a sus campos, se me complicó mucho inicialmente. También es cierto, que este esfuerzo me ha ayudado en los ejercicios posteriores.

**Ejercicio 3**

Problema: Escribir un programa en C que pida al usuario un total de 10 palabras (máximo 20 caracteres) y luego las imprima todas ellas en pantalla por orden alfabético.

Resolución del problema: Aunque seguramente este ejercicio se pueda resolver de otra manera, dado que en el ejercicio anterior tuvo problemas para entender el manejo de estructuras y sobre todo el uso de punteros, investigando encontré varias referencias a como resolver el problema utilizando una estructura para las palabras

Comentarios: