PRÁCTICA 3: Recursividad

Objetivo: Diseñar e implementar subprogramas recursivos. Aprender a realizar trazas de los mismos y comprender cómo se ejecutan *(Fases en la ejecución de un subprograma recursivo y utilización del Stack).*

Enunciado: ***(leálo completo ANTES de afrontar la resolución de la práctica)***

Vamos a trabajar con dos vectores que almacenan los siguientes datos enteros:

- Vector V1: 1 12 23 28 35 Vector V2: 15 25 32 70 91

El programa mostrará en pantalla, de forma repetitiva, el siguiente menú de opciones:

```
                  MENÚ
               ===========

1. Producto de los Elementos del Vector v1
2. Invertir dígitos de un elemento del vector v1
3. Contar Pares en el vector v1
4. Combinar Vectores v1 y v2 (sin ordenar)
5. Combinar Vectores v1 y v2 (ordenado ascendente) (versión iterativa)
6. Combinar Vectores v1 y v2 (ordenado ascendente) (versión recursiva)
0. Salir

Elija opción:
```

La salida en pantalla que se obtendrá al ejecutar cada una de las opciones del menú se muestra a continuación:

- Opción 1: ***(subprograma recursivo producto*ElementosVector()*)***

```
Vector V1: 1 12 23 28 35

El producto de los elementos del vector es: 270480

Pulse <intro> para continuar...
```

- Opción 2: ***(subprograma recursivo invertirDigitos())***

```
Vector V1: 1 12 23 28 35

¿Qué número desea considerar?: 100

Este número no se encuentra en el vector v1..

Pulse <intro> para continuar...
```

```
Vector V1: 1 12 23 28 35

¿Qué número desea considerar?: 23

El número invertido es: 32

Aviso : El vector no quedará modificado

Pulse <intro> para continuar...
```

- Opción 3: ***(subprograma recursivo contarParesVector())***

```
Vector V1: 1 12 23 28 35

Cantidad de pares en el vector: 2

Pulse <intro> para continuar...
```

- Opción 4: ***(subprograma recursivo combinarVectores())***

```
Vector V1: 1 12 23 28 35

Vector V2: 15 25 32 70 91

Vector resultante combinado: 1 15 12 25 23 32 28 70 35 91

Pulse <intro> para continuar...
```

- Opción 5: ***(subprograma iterativo combinarOrdenar_ITER())***

```
Vector V1: 1 12 23 28 35

Vector V2: 15 25 32 70 91

Vector combinado y ordenado ascendentemente (versión iterativa):

1 12 15 23 25 28 32 35 70 91

Pulse <intro> para continuar...
```

- Opción 6: ***(subprograma recursivo combinarOrdenar_REC())***

```
Vector V1: 1 12 23 28 35

Vector V2: 15 25 32 70 91

Vector combinado y ordenado ascendentemente (versión recursiva):

1 12 15 23 25 28 32 35 70 91

Pulse <intro> para continuar...
```

Muy importante:

- Los vectores V1 , V2 se mostrarán desde el main cuando sea necesario
- El subprograma recursivo/iterativo indicado en cada caso debe realizar únicamente la tarea considerada *(producto de los elementos del vectors, calcular la suma de los dígitos de un número determinado, contar el número de elementos pares que hay en un vector, etc..)*
- Los resultados concretos *(producto de los elementos de un vector, etc)* se mostrarán desde el main después de que haya finalizado la llamada al subprograma correspondiente.
- Cuide el estilo de programación: comentarios en cada subprograma, etc.

DOCUMENTACIÓN A ENTREGAR

Será preciso realizar una **memoria** organizada de la siguiente forma:

1.- Portada: Indicando Título de la práctica, Grupo de prácticas, Autor/es, curso 1º y fecha de entrega

2.- La memoria debe contener lo siguiente de cada una de las opciones del menú:

a) Para la opción 5 del menú: análisis del problema *(describa brevemente el proceso a llevar a cabo para alcanzar la solución)*

b) Para cada una de las opciones 1, 2, 3, 4 y 6 del menú:

- Análisis del problema
- Diseño del algoritmo recursivo => identifique caso/s base, caso/s recursivo/s *(ley de recurrencia)*
- ¿Qué tipo/s de recursividad se implementa?
- Analice el código fuente de la aplicación e identifique - para cada uno de los subprogramas recursivos implementados - qué operaciones se llevan a cabo en cada una de las distintas etapas que se contemplan en la ejecución de un subprograma recursivo (desplegado – caso base - plegado). Debe incluir en la memoria el código fuente del subprograma recursivo considerado y marcar lo que se ejecuta en cada etapa.

c) Incluya en la memoria una traza de ejecución de los subprogramas recursivos considerados en las opciones 2 y 3 del menú (considere el caso concreto que se muestra en los cuadros negros de ejemplo)

3.- Pruebas de ejecución del programa (Capturas de pantalla de ejecución de cada una de las opciones del menú)

ENTREGA DE LA PRÁCTICA

La entrega de la práctica se realizará a través del campus virtual de la asignatura teniendo en cuenta las instrucciones que allí se detallan. Deberá entregarse un fichero comprimido (.zip) que contenga la memoria completa de la práctica y el código fuente del programa implementado (consultad el campus virtual).

- Fecha de publicación de la práctica: 3 de Mayo de 2024
- Fecha de entrega de la práctica: 26 de Mayo de 2024, 22 h. – Pasada esta fecha no se admitirán nuevas entregas en la convocatoria ordinaria.

EVALUACIÓN:

* En la evaluación de la práctica se tendrá especialmente en cuenta la documentación interna, la calidad de la estructuración del programa en subprogramas y la adecuación a los contenidos teóricos estudiados en la asignatura.
* Aunque las prácticas se presentan en parejas, la calificación es individual: Cada alumno debe responsabilizarse de su participación en la elaboración de las mismas.

MUY IMPORTANTE:
En el caso en que se detecte la copia/plagio de una práctica, todos los alumnos implicados obtendrán una calificación "SUSPENSO" (0.0) en dicha práctica en la convocatoria Ordinaria y deberán entregar otra práctica distinta y de mayor complejidad en la convocatoria Extraordinaria.
- Se recuerda que es preciso aprobar las prácticas para poder aprobar la asignatura.