METODOLOGÍA DE LA PROGRAMACIÓN

PRÁCTICA 3

GRUPO 30 | Daniel Atanasov y Raúl Sanz

PRACTICA 3 | Recursividad

PROFESORA | María del Pilar Grande González y Berta San Isidro

ÍNDICE

1. Contar vocales en V1
    1.1. Análisis del problema
    1.2. Diseño del algoritmo recursivo
    1.3. Tipo de recursividad
    1.4. Análisis del código fuente
    1.5. Traza de ejecución
2. Eliminar todas las consonantes de V2
    2.1- Análisis del problema
    2.2- Diseño del algoritmo recursivo
    2.3- Tipo de recursividad
    2.4- Análisis del código fuente
3. Intercalar letras de V1 y V2 en un nuevo string V3
    3.1- Análisis del problema
    3.2- Diseño del algoritmo recursivo
    3.3- Tipo de recursividad
    3.4- Análisis del código fuente
4. Duplicar cada letra de V1
    4.1- Análisis del problema
    4.2- Diseño del algoritmo recursivo
    4.3- Tipo de recursividad
    4.4- Análisis del código fuente
5. Convertir mayúsculas de V1 y V2 a minúsculas (versión iterativa) – Análisis del problema
6. Convertir mayúsculas de V1 y V2 a minúsculas (versión recursiva)
    6.1- Análisis del problema
    6.2- Diseño del algoritmo recursivo
    6.3- Tipo de recursividad
    6.4- Análisis del código fuente
    6.5- Traza de ejecución
7. Ordenar alfabéticamente la cadena V1+V2
    7.1- Análisis del problema
    7.2- Diseño del algoritmo recursivo
    7.3- Tipo de recursividad
    7.4- Análisis del código fuente
8. Pruebas de ejecución

1. Contar Vocales de V1

1.1- Análisis del problema

En este caso, nos enfrentamos al problema de recorrer una cadena de caracteres y contar cuántas de estas letras son vocales.

1.2- Diseño del algoritmo recursivo

El caso base para este subprograma se encuentra cuando la cadena con la que estamos trabajando se termina, es decir, cuando llegamos al término '\0' que indica ese final de cadena.

El caso recursivo analiza si el carácter es una vocal o no, en case de que sí lo sea suma 1 a la variable que contabiliza las vocales y, en caso contrario no suma nada.

1.3- Tipo de recursividad

Se trata de una recursividad lineal no final ya que después de la llamada recursiva todavía tiene que seguir realizando su trabajo.

1.4- Análisis del código fuente

En las etapas de este programa recursivo se llevan a cabo las siguientes operaciones: durante el desplegado se evalúa la cadena en busca de vocales comprobando carácter a carácter hasta llegar al caso base descrito anteriormente, entonces comienza la fase de plegado donde cada llamada recibe el resultado de la llamada anterior sumando 0 o 1 en función de si se trataba de una vocal y retorna sin modificar la cadena.

1.5- Trazas de ejecución

Llamada	Carácter	¿Vocal?	Devuelve
0	m	No	0+…
1	u	Si	1+…
2	r	No	0+…
3	c	No	0+…
4	i	Si	1+…
5	e	Si	1+…
6	l	No	0+…
7	a	Si	1+…
8	g	No	0+…
9	o	Si	1+…
10	'\0'	-	0

Resultado final: 5 vocales

2. Eliminar Todas las Consonantes de V2

2.1- Análisis del problema

En este subprograma tenemos que recorrer una cadena de caracteres en busca de las consonantes para crear un nuevo string que contenga sólo las vocales de la cadena original, es decir, eliminamos las consonantes.

2.2- Diseño del algoritmo recursivo

El caso base para este subprograma se encuentra cuando la cadena con la que estamos trabajando se termina, es decir, cuando llegamos al término '\0' que indica ese final de cadena.

En cuanto al caso recursivo, en caso de que el carácter sea una vocal la copia en el nuevo string y avanza para seguir buscando las vocales.

2.3- Tipo de recursividad

Se trata de una recursividad lineal final ya que la llamada recursiva es la última operación que se ejecuta en el subprograma.

2.4- Análisis del código fuente

En las etapas de este programa recursivo se llevan a cabo las siguientes operaciones: durante el desplegado se evalúa la cadena carácter a carácter en busca de vocales que se copian en otro string hasta llegar al caso base descrito anteriormente, entonces comienza la fase de plegado en la cual no se realizan operaciones adicionales al regresar.

3. Intercalar Letras de V1 y V2 en un Nuevo String V3

3.1- Análisis del problema

El problema al que nos enfrentamos con este subprograma es el de recorrer dos cadenas de caracteres para poner sus letras intercaladamente en un nuevo string. En el caso de que alguna termine antes que la otra, continúa con los caracteres restantes de la otra.

3.2- Diseño del algoritmo recursivo

El caso base en este algoritmo recursivo se encuentra cuando ambas cadenas terminan, es decir, cuando ambas han llegado a su último término: '\0'.

El caso recursivo copia el carácter correspondiente de cada cadena y avanza para seguir copiando los caracteres.

3.3- Tipo de recursividad

La recursividad en este caso es de tipo lineal final, ya que todas las operaciones se realizan antes de la llamada recursiva, la cual va al final.

3.4- Análisis del código fuente

Durante las etapas de este subprograma se realizan las siguientes operaciones: en el despliegue se copian las letras correspondientes hasta llegar al caso base; entonces empieza el plegado donde no se realiza ninguna operación.

4. Duplicar Cada Letra de V1

4.1- Análisis del problema

En este subprograma hemos tenido que recorrer una cadena de caracteres letra a letra para duplicar cada una de ellas en un nuevo string.

4.2- Diseño del algoritmo recursivo

En este algoritmo recursivo el caso base se encuentra cuando termina la cadena de caracteres, en el término '\0'. Mientras que el caso recursivo consiste en copiar por duplicado cada carácter del string en uno nuevo y seguir hasta haber duplicado todos los caracteres.

4.3- Tipo de recursividad

La recursividad en este subprograma vuelve a ser de tipo lineal final porque después de la llamada no se realizan más operaciones.

4.4- Análisis del código fuente

En este subprograma las fases son las siguientes: fase de desplegado, se recorre la cadena de caracteres letra a letra copiando cada uno dos veces en un nuevo string hasta llegar al caso base, el término '\0' final de la cadena, para terminar con la fase de plegado donde no se realizan operaciones nuevas.

5. Convertir Mayúsculas de V1 y V2 a Minúsculas (Versión Iterativa) – Análisis del Problema

En este subprograma iterativo hemos tenido que recorrer dos cadenas de caracteres mediante un bucle for para cada una en busca de letras en mayúscula para transformarlas en letras minúsculas. Antes de los bucles for, hemos establecido dos variables con la longitud de cada cadena para saber cuándo tenían que finalizar los bucles for. Al terminar cada bucle, devuelve la cadena, pero con todas sus letras en minúscula.

Para la detección de las mayúsculas y su transformación en minúsculas empleamos isupper y tolower de la librería ctype.h.

6. Convertir Mayúsculas de V1 y V2 a Minúsculas (Versión Recursiva)

6.1- Análisis del problema

En este caso hemos tenido que recorrer dos cadenas de caracteres por separado en búsqueda de caracteres en mayúscula para transformarlos en minúscula y devolver cada cadena con todas sus letras en minúscula. Para esto volvemos a utilizar la librería ctype.h.

6.2- Diseño del algoritmo recursivo

El caso base se encuentra en el final de la cadena nuevamente. Mientras que el caso recursivo procesa la cadena V1 hasta que termina con ella y entonces procesa V2. Mientras procesa cada cadena, busca las mayúsculas para transformarlas y avanzar.

6.3- Tipo de recursividad

La recursividad en este subprograma es de tipo lineal final.

6.4- Análisis del código fuente

En las etapas de este programa recursivo se llevan a cabo las siguientes operaciones: durante el desplegado se evalúa cada cadena carácter a carácter por separado en busca de mayúsculas, las cuales se transforman en minúscula y avanza hasta llegar al caso base del final de cadena, entonces comienza la fase de plegado en la cual no se realizan operaciones adicionales al regresar. Después de todo esto, devuelve la cadena transformada en la cadena que recibió.

6.5- Trazas de ejecución

Traza de ejecución para la cadena V1 = “MurcIeLago”

Llamada	Carácter	¿Mayúscula?	Resultado
0	“M”	Si	m
1	“u”	No	mu
2	“r”	No	mur
3	“c”	No	murc
4	“I”	Si	murci
5	“e”	No	murcie
6	“L”	Si	murciel
7	“a”	No	múrciela
8	“g”	No	murcielag
9	“o”	No	murcielago
10	‘\0’	-	FIN

Traza de ejecución para la cadena V2 = “ORdenADOr”

Llamada	Carácter	¿Mayúscula?	Resultado
0	“O”	Si	o
1	“R”	Si	or
2	“d”	No	ord
3	“e”	No	orde
4	“n”	No	orden
5	“A”	Si	ordena
6	“D”	Si	ordenad
7	“O”	Si	ordenado
8	“r”	No	Ordenador
9	‘\0’	-	FIN

7. Ordenar Alfabéticamente la Cadena V1+V2

7.1- Análisis del problema

En este subprograma hemos tenido que concatenar dos cadenas en una nueva y, seguidamente, ordenarlas alfabéticamente mediante un algoritmo recursivo.

7.2- Diseño del algoritmo recursivo

El caso base se encuentra cuando la longitud de la cadena es menor o igual que 1, esto es debido a que ya no quedaría nada por ordenar.

El caso recursivo compara los caracteres por pares adyacentes ordenándolos así poco a poco con cada ejecución, luego se llama nuevamente al algoritmo con una longitud menor.

7.3- Tipo de recursividad

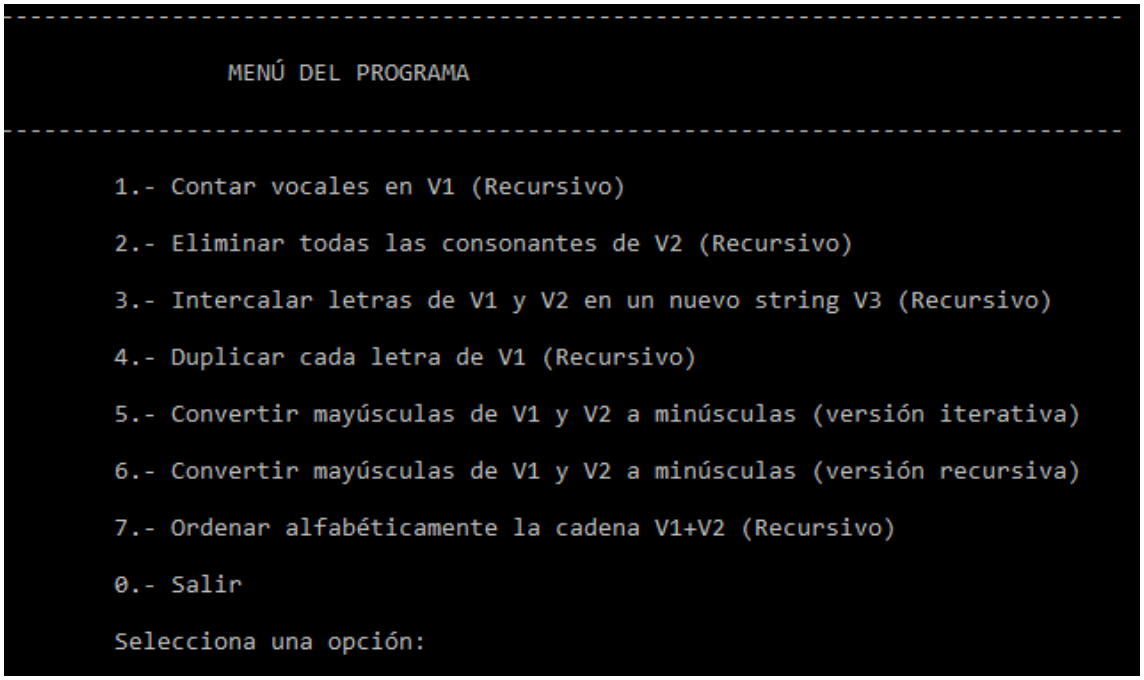
La recursividad en este subprograma es del tipo lineal no final, porque después de la llamada todavía tiene que hacer un último bucle for, entonces, si no se producen cambios, ya termina la ejecución del algoritmo.

7.4- Análisis del código fuente

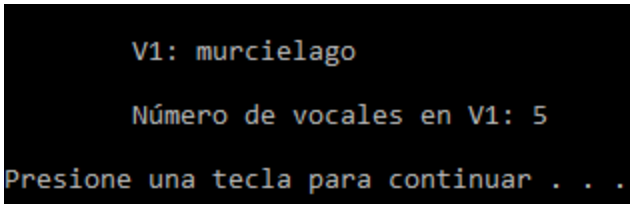
En este caso, las operaciones que se llevan a cabo durante las distintas fases son las siguientes: durante el despliegue el algoritmo realiza llamadas recursivas reduciendo la longitud de la cadena y ordenándola hasta que llega al caso base, donde finalizan las llamadas y comienza la fase de plegado, donde no se realizan operaciones nuevas.

8. Pruebas de Ejecución

Menú de opciones del programa:



Opción 1 – Contar Vocales:



Opción 2- Eliminar Consonantes:

```
        V2 original: ordenador

        V2 sin consonantes: oeao

Presione una tecla para continuar . . .
```

Opción 3 – Intercalar letras:

```
        V1: murcielago

        V2: ordenador

        Cadena intercalada V3: mourrdceinealdaogro

Presione una tecla para continuar . . .
```

Opción 4 – Duplicar letras:

```
        V1: murcielago

        V1 duplicado: mmuurrcciieellaaggoo
Presione una tecla para continuar . . .
```

Opción 5 – Mayúsculas a Minúsculas Iterativo:

```
        V1: MurcIeLago

        V2: ORdenADOr

        V1 en minusculas: murcielago

        V2 en minusculas: ordenador

Presione una tecla para continuar . . .
```

Opción 6 – Mayúsculas a Minúsculas Recursivo:

```
        V1: MurcIeLago

        V2: ORdenADOr

        V1 en minusculas: murcielago

        V2 en minusculas: ordenador

Presione una tecla para continuar . . .
```

Opción 7 – Ordenar Alfabéticamente:

```
        V1: murcielago

        V2: ordenador

        Cadena combinada ordenada: aacddeegilmnooorrru

Presione una tecla para continuar . . .
```

Opción 0 – Salir:

```
        ¡Gracias por usar nuestro programa!


Process returned 0 (0x0)   execution time : 219.376 s
Press any key to continue.
```