Ayuda para la programación en C

Programa 1: Devolución de moneda. (3 puntos)

Realizar un programa C completo (main + función) que simule el cálculo del número de monedas a devolver en una máquina de cobro de servicios similar a las que podemos encontrar en el Metro o en un Parking.

La máquina según la petición del servicio del cliente, calcula el **importe** del servicio tras lo cual el cliente ingresa una cantidad de dinero (**cant_intro**) que deberá ser suficiente para realizar el cobro. A continuación, calculará el importe a **devolver** y dicha devolución la hará siempre con monedas de manera que devuelva el menor número posible de ellas, es decir, empezará entregando monedas desde la de mayor a la de menor valor.

Las monedas serán de 2.00, 1.00, 0.50, 0.20, 0.10, 0.05, 0.02 y 0.01 euros.

Supondremos que la máquina siempre tiene suficientes monedas para devolver el cambio.

El programa realizará el siguiente proceso:

1. Inicializará el vector **valores** con los correspondientes a las monedas de curso legal en céntimos.
2. Leerá y validará por teclado el **importe** del servicio que será un valor real y positivo en euros (es necesario validar).
3. Leerá y validará por teclado la cantidad de dinero introducida por el cliente (**cant_intro**) que será un valor real en euros y, mayor o igual al **importe** a pagar (es necesario validar).
4. Calculará el importe a **devolver** y escribirá éste en pantalla.
5. Llamará a la función **DesgloseDeMoneda** que recibe el valor de la cantidad a **devolver** y dos vectores, el primero **valores** de tipo entero de 8 elementos con los valores de las monedas en céntimos y el segundo **monedas** también de tipo entero de 8 elementos para devolver el número de monedas de cada valor al main.
6. En el main se escribirá el desglose de las monedas correspondientes a partir del vector **monedas**.

La función **DesgloseDeMoneda** calculará el número de monedas de cada valor que hay que entregar en la devolución. Su prototipo será:

```
void DesgloseDeMoneda(float devolver, int valores[], int monedas[]);
```

En el main se puede inicializar el vector **valores** en la declaración del mismo de la siguiente forma:

```
int valores[8] = {200, 100, 50, 20, 10, 5, 2, 1};
```

que sería equivalente a:

```
int valores[8];
valores[0] = 200;
...
valores[7] = 1;
```

Nota: Si el cliente introduce el importe exacto el programa no deberá llamar a la función y se indicará con un mensaje en pantalla desde el main.

Ejemplo ejecución:

```
Introducir el importe del servicio: 5.07
Introducir la cantidad para el cobro: 5
La cantidad introducida debe ser mayor que 5.07 euros

Introducir la cantidad para el cobro: 10

La cantidad a devolver es de 4.93 euros
2 moneda(s) de 2.00 euros
1 moneda(s) de 0.50 euros
2 moneda(s) de 0.20 euros
1 moneda(s) de 0.02 euros
1 moneda(s) de 0.01 euros
```

Programa 2: Romanos. (3.5 puntos)

Ayuda para la programación en C

Como bien es sabido, los legionarios romanos para entrar en batalla se agrupaban en forma geométrica (normalmente un rectángulo) y se cubrían con escudos.
Hubo un general que estableció que la mejor figura para la formación no era la rectangular sino la cuadrada, de forma que el número de filas y columnas de legionarios coincidía. Por tanto, el problema era decidir en cuántas formaciones (y de qué tamaño) debía separar su ejército.
La manera de organizar a los soldados en las distintas formaciones cuadradas es como sigue; Se comienza haciendo el cuadrado más grande posible con los legionarios. Con los que le quedan libres se vuelve a repetir la operación, y así hasta que no quedan legionarios que formar (3 o menos).

Por ejemplo, si el número de legionarios en el ejército es 35, la formación consistirá en un cuadrado de 25 legionarios (5×5), otro de 9 (3×3) y un soldado (1×1).
Los legionarios que ocupan las posiciones interiores necesitan un solo escudo, mientras que los que ocupaban los flancos llevaban dos, si no estaban en las esquinas y tres en caso contrario. Al final si sólo quedara 1, 2 ó 3 soldados llevarían tres escudos cada uno.
Para la formación anterior el número de escudos es como sigue:

```
3 2 2 2 3      3 2 3
2 1 1 1 2      2 1 2      3
2 1 1 1 2      3 2 3
2 1 1 1 2
3 2 2 2 3
```

1ª formación 2ª formación 3ª formación

A continuación, se muestra otro ejemplo en el que el número de legionarios del ejército es 19, por tanto, la formación consistirá en un cuadrado de 16 legionarios (4×4) y tres soldados.

```
3 2 2 3      3 3
2 1 1 2      3
2 1 1 2
3 2 2 3
```

1ª formación 2ª formación

El programa debe solicitar el número de legionarios (como máximo 1000 y como mínimo 0, hay que validarlo mostrando mensaje de error en su caso) y calcular y mostrar el total de escudos de ejército.

Nota: No se deben usar matrices, ni dibujar nada.

Ayuda para la programación en C

Ejemplo ejecución 1º:

```
Introduzca el número de legionarios: 35
El total de escudos será: 69
```

Explicación: (no es necesario que lo muestre el programa)

Primera formación
Cuadrado de dimensión 5
4 legionarios en esquinas
12 legionarios en los flancos
9 legionarios en el interior
Total Primera Formación: 45

Segunda formación
Cuadrado de dimensión 3
4 legionarios en esquinas
4 legionarios en los flancos
1 legionarios en el interior
Total Segunda Formación: 21

Tercera formación
1 legionario
Total Tercera Formación: 3
TOTAL= 45 + 21 + 3 = 69 escudos

Ejemplo ejecución 2º:

Introduzca el número de legionarios: 19
El total de escudos será: 41

Explicación: (no es necesario que lo muestre el programa)

Primera formación
Cuadrado de dimensión 4
4 legionarios en esquinas
8 legionarios en los flancos
4 legionarios en el interior
Total Primera Formación: 32

Segunda formación
3 legionarios
Total Tercera Formación: 9
TOTAL= 32 + 9 = 41 escudos

Ayuda para la programación en C

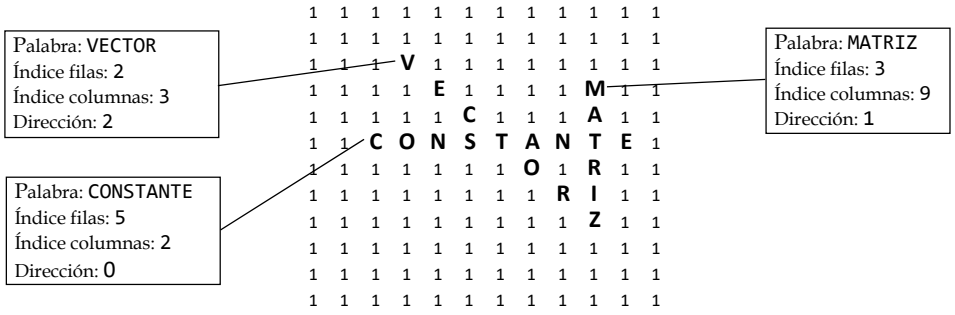
Programa 3: Sopa de letras. (3.5 puntos)

Para un programa que permite la generación de sopas de letras, se deben **codificar únicamente las siguientes funciones:** `Verificar_Long_Palabra()` y `Comprobar_y_Escribir_Palabra()`. Estas funciones serán descritas más adelante en el enunciado.

El programa referido comienza (en `main` y por tanto no hay que codificarlo) generando una matriz NxN de tipo `char` inicializada a '1' en todas sus posiciones, donde N es una constante simbólica. Una vez inicializada la matriz, las palabras que se escriban en la misma podrán ser de los siguientes tipos:

- Horizontal: Palabras que se escribirán de izquierda a derecha en las filas de la matriz.
- Vertical: Palabras que se escribirán en vertical y hacia abajo en las columnas de la matriz.
- Diagonal: Palabras que se escribirán en diagonal descendente y hacia la derecha en la matriz.

Además de la palabra que se introducirá en la matriz, se deberán conocer los índices de la posición (fila y columna) donde comenzará la escritura de la misma y la dirección en la que se pondrá la palabra (**Horizontal=0, Vertical=1, Diagonal=2**). Ejemplo de palabras en una matriz con dimensión N=12:



Una vez introducidas las palabras deseadas, el programa completaría la sopa de letras sustituyendo los '1' que queden en la matriz por letras de forma aleatoria.

Para el programa descrito, **se deberán codificar únicamente** las siguientes funciones:

`Verificar_Long_Palabra()`:

Esta función recibe la **longitud** de la palabra a escribir en la matriz, el **vector** con los índices de fila (primer elemento del vector) y columna (segundo elemento del vector) desde donde se comenzará a escribir la palabra y, por último, la **dirección** en la que se escribirá la palabra (0, 1 ó 2).

La función comprobará que la longitud de la palabra que se quiere escribir, a partir de los índices indicados en el vector `indices`, no excede los límites de la matriz. El prototipo de la función es:

```
int Verificar_Long_Palabra(int long_palabra, int indices[], int direccion);
```

Si la palabra no excede los límites de la matriz la función devolverá 0. En caso de excederlos, devolverá -1.

`Comprobar_y_Escribir_Palabra()`:

Esta función recibe la **matriz** con las palabras en la sopa de letras escritas hasta el momento, la cadena de caracteres con la **palabra** a escribir en la matriz, el **vector** con los índices de la posición donde comienza la palabra, la **dirección** en la que se escribirá la palabra y la **longitud** de la palabra.

El prototipo de la función es:

```
int Comprobar_y_Escribir_Palabra(char matriz[][N], char palabra[], int indices[],
int direccion, int long_palabra);
```

La función primero comprobará que la palabra recibida se escribiría únicamente en posiciones de la matriz en las que su valor sea `'1'` o, en caso tener la matriz ya escrita una letra en una determinada posición, ésta sea coincidente con la letra correspondiente de la palabra a copiar en la matriz. Ejemplo: En la figura anterior, al querer escribir la palabra `"MATRIZ"` y haber ya escritas dos palabras, en la posición de la fila 5 y la columna 9 encuentra una letra ya escrita, pero como coincide con la `'T'` que se quiere escribir, es posible añadir la nueva palabra en la matriz.

Una vez que se ha determinado que la palabra puede ser escrita en la matriz se procederá, dentro de la misma función, a su escritura. En caso de haberse escrito la palabra correctamente, la función devolverá 0. En caso de no haberse podido escribir devolverá -1.