Índice

Función sopaDeLetras()

llenarAbecedario(abecedario)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***abecedario*** el cual, si bien no está definido, es del ***tipo carácter***. El mismo fue previamente dimensionado de la siguiente forma:

```
cantidadLetrasAbecedario←27
Dimensionar abecedario[cantidadLetrasAbecedario]
```

➢ *Retorno:*

Esta función no tiene un valor de retorno.

➢ *Funcionamiento:*

Al utilizar el pasaje por referencia, la función llena al vector ***abecedario*** con las 27 letras que componen al abecedario en español (una en cada índice del vector).

llenarNumeros(vNumeros)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***vNumeros*** el cual, si bien no está definido, es del ***tipo carácter***. El mismo fue previamente dimensionado de la siguiente forma:

```
cantidadNumeros←10
Dimensionar vNumeros[cantidadNumeros]
```

➢ *Retorno:*

Esta función no tiene un valor de retorno.

➢ *Funcionamiento:*

Al utilizar el pasaje por referencia, la función llena al vector ***vNumeros*** con los 10 dígitos que conforman al sistema decimal. Los mismos fueron almacenados como carácter a fin de evitar errores en el programa[*1].

inicializarMatriz(grilla, filasGrilla, columnasGrilla)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia a la matriz ***grilla*** la cual, si bien no está definida, es del ***tipo carácter***. La misma fue previamente dimensionada de la siguiente forma:

```
filasGrilla←15
columnasGrilla←15
Dimensionar grilla[filasGrilla,columnasGrilla]
```

A su vez recibe por valor los argumentos filasGrilla y columnasGrilla.

➢ *Retorno:*

Esta función no tiene una variable de retorno.

➢ *Funcionamiento:*

Al utilizar el pasaje por referencia, la función llena la matriz ***grilla*** con guiones (“-”). Los mismos fueron almacenados con el fin de establecer un estado inicial conocido en la matriz.

inicializarVector(familiaDePalabras, cantidadMaximaPalabras)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***familiaDePalabras*** el cual, si bien no está definido, es del ***tipo carácter***. El mismo fue previamente dimensionado de la siguiente forma:

```
cantidadMaximaPalabras←10
Dimensionar familiaDePalabras[cantidadMaximaPalabras]
```

➢ *Retorno:*

Esta función no tiene una variable de retorno.

➢ *Funcionamiento:*

Al utilizar un pasaje por referencia, la función llena el vector ***familiaDePalabras*** con puntos (“.”). Los mismos fueron almacenados con el fin de establecer un estado inicial conocido en el vector.

mostrarMenuFamilias()

➢ *Argumento/s:*

Esta función no recibe argumentos.

➢ *Retorno:*

Esta función no tiene variable de retorno.

➢ *Funcionamiento:*

La función muestra el menú de familias/categorías de palabras sobre las que el usuario puede hacer selección en la función seleccionFamilia().

seleccionFamilia()

➢ *Argumento/s:*

Esta función no recibe argumentos.

➢ *Retorno:*

Esta función tiene retorno la variable ***codigoFamilia***.

➢ *Funcionamiento:*

La función consta de un ciclo do-while el cual cuenta con un contador de ciclos (***contadorCiclos***) con el fin de actualizar el texto en caso de una primera selección incorrecta.
En cada ciclo la función hace uso de la función ***mostrarMenuFamilias()*** para

darle contexto al usuario sobre cuales son las opciones válidas disponibles.
Dentro del ciclo la entrada del usuario, almacenada en la variable ***codigo***, es manipulada como una cadena de caracteres con el fin de evitar que el ingreso de un carácter o cadena de caracteres termine con el programa por un *error 120*.[1] Por lo cual las condiciones para romper el ciclo, son las opciones válidas del menú convertidas a caracteres (valores numéricos entre 0 y 5).
Una vez finalizado el ciclo le asignamos a ***codigoFamilia***, mediante la función ConvertirANumero, el valor numérico de la variable ***codigo***.

animacionTexto(palabra, simbolo, cantSimbolos)

➢ *Argumento/s:*

Esta función recibe como argumentos por valor a las variables ***palabra***, ***simbolo*** y ***cantSimbolos***.

➢ *Retorno:*

Esta función no tiene variable de retorno.

➢ *Funcionamiento:*

Esta función consta de un ciclo for, en el que inicialmente se limpia la pantalla y al finalizar se agrega un delay. Desde el segundo ciclo se actualiza el contenido de ***palabra*** concatenandolo con el ***símbolo*** una cantidad de veces ***cantSimbolos***.
El resultado visual es una animación de la palabra en conjunto con los símbolos.

validarCantidadPalabras(cantidadMaximaPalabras, vectorNumeros)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***vectorNumeros*** y por valor a la variable ***cantidadMaximaPalabras***.

[1] El error 120 ocurre al no haber coincidencia entre el tipo de variable y el valor ingresado.

Este tratamiento fue realizado en múltiples funciones. Por lo que, a fin de evitar repeticiones, a partir de este punto se hará referencia al caso con el indicador [*1].

➢ *Retorno:*

Esta función retorna la variable ***palabrasUtilizadas***.

➢ *Funcionamiento:*

Esta función está formada por un ciclo do-while cuya condición de salida consiste en que el ingreso del usuario sea un número menor o igual a la variable ***cantidadMaximaPalabras***.
Con el fin de validar el ingreso del usuario, ***entradaUsuario*** recibe el tratamiento mencionado en la nota 1. A su vez, creamos una variable ***entradaValida*** a la cual le asignamos el valor inicial en 0.
Mediante dos ciclos for se compara el valor de cada carácter de ***entradaUsuario*** con los valores almacenados en el vector ***vectorNumeros***. En el caso de existir una coincidencia se incrementa en 1 el valor de ***entradaValida***.
En esta instancia existen solo cuatro posibilidades:

A. ***entradaValida***=Longitud(***entradaUsuario***)
    1. ***cantidadPalabrasUsuario***>***cantidadMaximaPalabras***
    2. ***cantidadPalabrasUsuario***=0
    3. *0*<***cantidadPalabrasUsuario***<=***cantidadMaximaPalabras***
B. ***entradaValida***≠Longitud(***entradaUsuario***)

A. Implica que todos los caracteres ingresados por el usuario fueron números, por lo que se le asigna a la variable de retorno ***cantidadPalabrasUsuario*** el valor de numérico de la variable ***entradaUsuario*** con ConvertirANumero(***entradaUsuario***).
    1 y 2. Se imprime en pantalla un mensaje informando al usuario el valor ingresado y los parámetros válidos para el ingreso.
    3. Se retorna el valor de numérico de la variable ***palabrasUtilizadas*** mediante el uso de ConvertirANumero(***palabrasUtilizadas***)
B. Implica que el usuario ingresó valores que no pertenecen al conjunto de los números naturales, por lo que se imprime en pantalla un mensaje informando al usuario el contenido de su ingreso y se le notifica que no es un número válido.

ingresarPalabraPersonalizada(cantidadMaximaLetras, abecedario)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***abecedario*** y por valor a la variable ***cantidadMaximaLetras***.

➢ *Retorno:*

Esta función retorna el valor ***palabraOK***.

➢ *Funcionamiento:*

Esta función está compuesta por un ciclo do-while en cual se itera mientras que se de alguna (o ambas) de la siguientes condiciones:

A. Longitud(**palabraOK**[2])>**cantidadMaximaLetras**
B. Que alguno de los caracteres que conforman a la cadena de caracteres de la variable **palabraOK**[2] no se encuentre en el vector ***abecedario***.

Fuera del ciclo establecemos un contador de ciclos (***contadorCiclos***) con un valor inicial en 0, y dentro del ciclo en última instancia se incrementa en 1 el valor actual de la variable ***contadorCiclos***. La misma es utilizada con el fin de establecer mensajes de error que informen al usuario sobre el contenido que se espera en su ingreso.

Con el fin de analizar el contenido de ***palabraOK***, el ciclo do-while cuenta con dos ciclos for anidados que comparan el valor de cada carácter de ***palabraOK*** mediante el uso de la función Subcadena() con cada valor del vector ***abecedario***. Si existen coincidencias se incrementa en 1 el valor de ***letrasOK***.

Al salir del ciclo do-while es decir, cuando la palabra ingresada por el usuario cumple con el largo máximo y con estar compuesta solo por letras, se retorna la variable ***palabraOK***.

cargarVector(familiaDePalabras, cantidadMaximaPalabras, largoMaximoPalabra, abecedario , vNumeros, op)

➢ *Argumento/s:*

Esta función recibe por referencia a los vectores ***familiaDePalabras***, ***abecedario*** y ***vNumeros*** los cuales fueron previamente inicializados en las funciones ***inicializarVector***, ***llenarAbecedario*** y ***llenarNumeros*** respectivamente.

Recibe por valor las variables ***cantidadMaximaPalabras***, ***largoMaximoPalabra*** y ***op*** las cuales fueron previamente inicializadas de la siguiente manera:

```
largoMaximoPalabra←10
op←seleccionFamilia()
```

[2] palabraOK es el ingreso inicial del usuario que puede, o no, ser válido.

Retorno:

Esta función retorna la variable ***cantidadPalabrasUsadas***.

➢ *Funcionamiento:*

Esta función consta de un switch en función de la variable ***op*** (previamente validada en la función ***seleccionFamilia***).

Los casos posibles para ***op*** son los siguientes:

- ➔ Entre 1-4: Carga el vector ***familiaDePalabras*** con la familia precargada que corresponda (Comidas, Tech, Vehículos o Animales).
- ➔ 5: Carga de palabras personalizada.

Previo al switch se establece el valor inicial de ***cantidadPalabrasUsuario*** en 10, (cantidad de palabras en el vector ***familiaDePalabras*** para las familias precargadas).
Es importante destacar que, el caso en el que la variable ***op*** es igual a 5 puede alterar el valor de ***cantidadPalabrasUsuario***. En este caso se hace uso de la función ***validarCantidadPalabras*** que retorna la cantidad de palabras utilizadas en la variable ***cantidadPalabrasUsuario,*** para posteriormente hacer uso de la función ***ingresarPalabraPersonalizada*** que itera dentro de un ciclo for una cantidad de veces ***cantidadPalabrasUsuario*** asignando las palabras ingresadas por el usuario al vector $familiaDePalabras_{[i]}$.
Finalmente la función retorna la variable ***cantidadPalabrasUsuario.***

palabraAVector(palabra, vectorAux)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***vectorAux*** y por valor a la variable ***palabra***.

➢ *Retorno:*

Esta función no tiene variable de retorno.

➢ *Funcionamiento:*

Consiste en un ciclo for en el cual cada carácter de la cadena ***palabra*** es asignado, mediante el uso de la función subcadena(***palabra***, ***i***, ***i***), al $vectorAux_{[i]}$.

orientarPalabras()

➢ *Argumento/s:*

Esta función no recibe argumentos.

➢ *Retorno:*

Esta función retorna la variable ***orientacion***.

➢ *Funcionamiento:*

Esta función le asigna a la variable de retorno ***orientacion*** un valor al azar entre 1 y 8. A continuación se adjunta la tabla de valores asociados a cada valor numérico:

Tabla de valores	
Valor Asociado	Orientación
1	Horizontal
2	Vertical
3	Diagonal Ascendente
4	Diagonal Descendente
5	Horizontal Invertida
6	Vertical Invertida
7	Diagonal Ascendente Invertida
8	Diagonal Descendente Invertida

palabraHorizontal(vectorFamilia$_{[i]}$, grilla, filas, columnas, i)

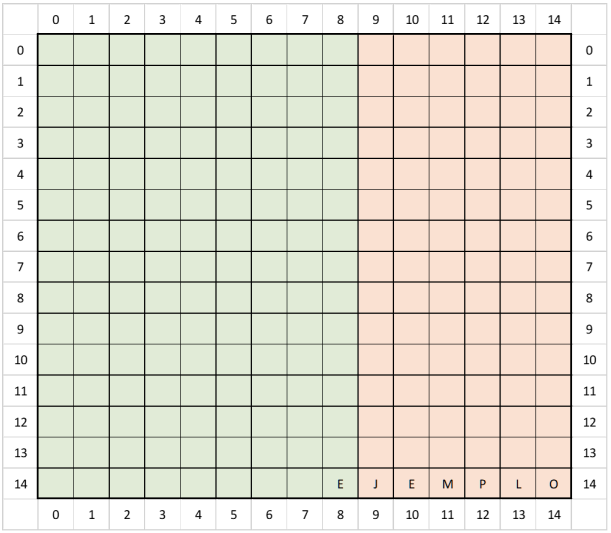
➢ *Argumento/s:*

Esta función recibe como argumento por referencia a la matriz ***grilla*** y por valor a las variables ***vectorFamilia$_{[i]}$***, ***filas***, ***columnas*** e ***i***.

➢ *Retorno:*

Esta función retorna la variable ***error***.

➢ *Funcionamiento:*



Características

- ***filaInicial*** = ***filaFinal*** ⇒ ***indiceFila*** = azar(***filas***)
- ***columnaInicialMáxima*** = ***columnas*** - Longitud(“EJEMPLO”) + 1
- ***indiceColumna*** = azar(***columnaInicialMáxima***)
- ***columnaFinal*** = ***indiceColumna*** + Longitud(“EJEMPLO”) - 1

Esta función consiste en un ciclo do-while cuya condiciones de salida son:

A. Que la colocación de la palabra sea válida.
B. Que la cantidad de ciclos haya alcanzado el valor máximo.

Fuera del ciclo se definen las variables ***contadorCiclos*** con valor inicial 0 y ***maximoCiclos*** cuyo valor inicial es el producto de ***filas*******columnas*** .
Al finalizar cada ciclo se incrementa en 1 el valor de ***contadorCiclos*** por lo que, si durante los ciclos previos a que ***contadorCiclos*** alcance el valor definido en ***maximoCiclos***, no se da la condición ***colocacionInvalida***=0 el ciclo finaliza con ***colocacionInvalida***=1.
El valor de ***colocacionInvalida***=0 es condición inicial al ingresar en cada ciclo y su valor se actualiza a ***colocacionInvalida***=1 en caso de que al evaluar, mediante el uso de un ciclo for que evalúa el contenido de la matriz en cada posición $grilla_{[indiceFila,i]}$ (con i incrementando en 1 su valor desde ***indiceColumna*** hasta ***columnaFinal***), encuentre valores distintos a “-” (valor de inicialización de la matriz) lo que implica la presencia de al menos un carácter que compone otra palabra.

Finalizado el ciclo for se evalúa el valor de la variable ***contadorCiclos***, si la misma es menor al valor máximo entonces se dimensiona al vector $\textbf{\textit{vectorAux}}_{[Longitud(palabra)]}$ y mediante el uso de la función ***palabraAVector*** se le asigna el contenido de $\textbf{\textit{vectorFamilia}}_{[i]}$ para luego repetir el ciclo for anteriormente mencionado, esta vez sin evaluar condiciones y asignando el cada posición de la matriz ***grilla*** previamente evaluada el contenido del $\textbf{\textit{vectorAux}}_{[indicePalabra]}$ en donde ***indicePalabra*** inicialmente vale 0 y aumenta en 1 su valor en cada iteración. La variable de retorno ***error*** se mantiene en 0.
En el caso de que no se cumpla la condición se retorna la variable ***error*** con el valor 1.

palabraVertical(vectorFamilia[i], grilla, filas, columnas, i)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia a la matriz ***grilla*** y por valor a las variables $\textbf{\textit{vectorFamilia}}_{[i]}$, ***filas***, ***columnas*** e ***i***.

➢ *Retorno:*

Esta función retorna la variable ***error***.

➢ *Funcionamiento:*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0																0
1																1
2																2
3																3
4																4
5																5
6																6
7																7
8															E	8
9															J	9
10															E	10
11															M	11
12															P	12
13															L	13
14															O	14
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	

Características

columnaInicial = ***columnaFinal*** ⇒ ***indiceColumna*** = azar(***columnas***)
filaInicialMaxima = ***filas*** - Longitud(“EJEMPLO”) + 1
indiceFila = azar(***filaInicialMaxima***)
filaFinal = ***indiceFila*** + Longitud(“EJEMPLO”) - 1

Esta función consiste en un ciclo do-while cuya condiciones de salida son:

C. Que la colocación de la palabra sea válida.
D. Que la cantidad de ciclos haya alcanzado el valor máximo.

Fuera del ciclo se definen las variables ***contadorCiclos*** con valor inicial 0 y ***maximoCiclos*** cuyo valor inicial es el producto de ***filas*******columnas*** .
Al finalizar cada ciclo se incrementa en 1 el valor de ***contadorCiclos*** por lo que, si durante los ciclos previos a que ***contadorCiclos*** alcance el valor definido en ***maximoCiclos***, no se da la condición ***colocacionInvalida***=0 el ciclo finaliza con ***colocacionInvalida***=1.
El valor de ***colocacionInvalida***=0 es condición inicial al ingresar en cada ciclo y su valor se actualiza a ***colocacionInvalida***=1 en caso de que al evaluar, mediante el uso de un ciclo for que evalúa el contenido de la matriz en cada posición $grilla_{[i,indiceColumna]}$ (con i incrementando en 1 su valor desde ***indiceFila*** hasta ***filaFinal***), encuentre valores distintos a "-" (valor de inicialización de la matriz) lo que implica la presencia de al menos un carácter que compone otra palabra.
Finalizado el ciclo for se evalúa el valor de la variable ***contadorCiclos***, si la misma es menor al valor máximo entonces se dimensiona al vector $vectorAux_{[Longitud(palabra)]}$ y mediante el uso de la función ***palabraAVector*** se le asigna el contenido de $vectorFamilia_{[i]}$ para luego repetir el ciclo for anteriormente mencionado, esta vez sin evaluar condiciones y asignando el cada posición de la matriz ***grilla*** previamente evaluada el contenido del $vectorAux_{[indicePalabra]}$ en donde ***indicePalabra*** inicialmente vale 0 y aumenta en 1 su valor en cada iteración. La variable de retorno ***error*** se mantiene en 0.
En el caso de que no se cumpla la condición se retorna la variable ***error*** con el valor 1.

palabraDiagonalAsc(vectorFamilia[i], grilla, filas, columnas, i)

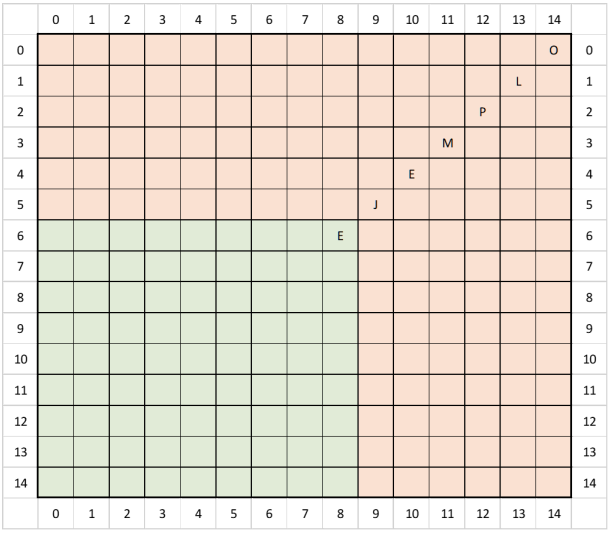
➢ *Argumento/s:*

Esta función recibe como argumento por referencia a la matriz ***grilla*** y por valor a las variables $vectorFamilia_{[i]}$, ***filas***, ***columnas*** e ***i***.

➢ *Retorno:*

Esta función retorna la variable ***error***.

➢ *Funcionamiento:*



Características

- ***filaInicialMinima***= Longitud(“EJEMPLO”) - 1
- ***indiceFila*** = azar(***filas*** - Longitud(“EJEMPLO”) + 1) + ***filaInicialMinima***
- ***filaFinal*** = ***indiceFila*** - Longitud(“EJEMPLO”) - 1
- ***columnaInicialMaxima*** = ***columnas*** - Longitud(“EJEMPLO”)
- ***indiceColumna*** = azar(***columnaInicialMaxima*** + 1)
- ***columnaFinal*** = ***indiceColumna*** + Longitud(“EJEMPLO”) - 1

Esta función consiste en un ciclo do-while cuya condiciones de salida son:

E. Que la colocación de la palabra sea válida.
F. Que la cantidad de ciclos haya alcanzado el valor máximo.

Fuera del ciclo se definen las variables ***columnaInicialMaxima*** como ***columnas***-***Longiutd***(***vectorFamilia***$_{[i]}$), ***filaInicialMinima*** como ***Longiutd***(***vectorFamilia***$_{[i]}$)-1 y ***contadorCiclos*** con valor inicial 0 y ***maximoCiclos*** cuyo valor inicial es el producto de ***filas*******columnas*** .
Al finalizar cada ciclo se incrementa en 1 el valor de ***contadorCiclos*** por lo que, si durante los ciclos previos a que ***contadorCiclos*** alcance el valor definido en ***maximoCiclos***, no se da la condición ***colocacionInvalida***=0 el ciclo finaliza con ***colocacionInvalida***=1.
El valor de ***colocacionInvalida***=0 es condición inicial al ingresar en cada ciclo y su valor se actualiza a ***colocacionInvalida***=1 en caso de que al evaluar, mediante el uso de un ciclo for que evalúa el contenido de la matriz en cada posición ***grilla***$_{[auxindiceFila,\ auxindiceColumna]}$ (con i incrementando en 1 su valor desde 0 hasta ***palabra***-1), encuentre valores distintos a “-” (valor de inicialización de la matriz) lo que implica la presencia de al menos un carácter que compone otra palabra.

En cada iteración del ciclo for decrementamos el valor de ***auxindiceFila*** en 1 e incrementamos el valor de ***auxindiceColumna*** en 1 para evaluar la siguiente posición de la matriz en la que queremos colocar la letra.
Finalizado el ciclo for se evalúa el valor de la variable ***contadorCiclos***, si la misma es menor al valor máximo si es asi se repite el ciclo for anteriormente mencionado, esta vez sin evaluar condiciones y asignando a cada posición de la matriz ***grilla***$_{[auxindiceFila,\ auxIndiceColumna]}$ previamente evaluada el contenido del ***Mayusculas(Subcadena(vectorFamilia$_{[i]}$,i,i))***. La variable de retorno ***error*** se mantiene en 0.
En el caso de que no se cumpla la condición se retorna la variable ***error*** con el valor 1.

palabraDiagonalDesc(vectorFamilia[i], grilla, filas, columnas, i)

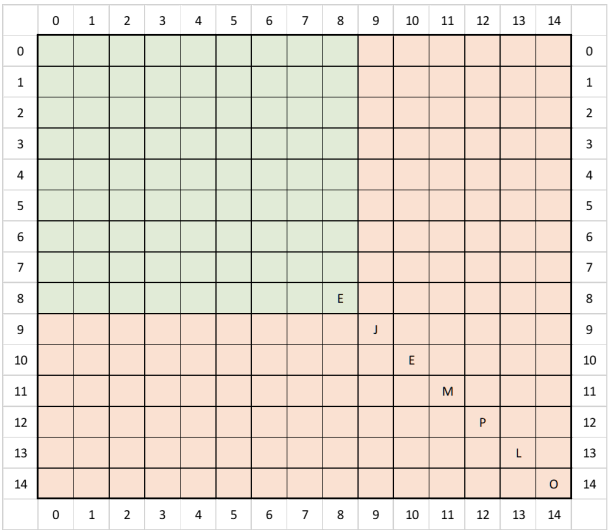
➢ *Argumento/s:*

Esta función recibe como argumento por referencia a la matriz ***grilla*** y por valor a las variables ***vectorFamilia$_{[i]}$***, ***filas***, ***columnas*** e ***i***.

➢ *Retorno:*

Esta función retorna la variable ***error***.

➢ *Funcionamiento:*



Características

- ***filaInicialMaxima*** = ***filas*** + 1 - Longitud(“EJEMPLO”)
- ***indiceFila*** = azar(***filaInicialMaxima***)
- ***filaFinal*** = ***indiceFila*** + Longitud(“EJEMPLO”) - 1

- ***columnaInicialMaxima*** = ***columnas*** + 1 - Longitud(“EJEMPLO”)
- ***indiceColumna*** = azar(***columnaInicialMaxima***)
- ***columnaFinal*** = ***indiceColumna*** + Longitud(“EJEMPLO”) - 1

Fuera del ciclo se definen las variables ***contadorCiclos*** con valor inicial 0 y ***maximoCiclos*** cuyo valor inicial es el producto de ***filas*******columnas*** .
Al finalizar cada ciclo se incrementa en 1 el valor de ***contadorCiclos*** por lo que, si durante los ciclos previos a que ***contadorCiclos*** alcance el valor definido en ***maximoCiclos***, no se da la condición ***colocacionInvalida***=0 el ciclo finaliza con ***colocacionInvalida***=1.
El valor de ***colocacionInvalida***=0 es condición inicial al ingresar en cada ciclo y su valor se actualiza a ***colocacionInvalida***=1 en caso de que al evaluar, mediante el uso de un ciclo for que evalúa el contenido de la matriz en cada posición $grilla_{[auxFila,auxColumna]}$ (con i incrementando en 1 su valor desde ***indiceFila*** hasta ***indiceFila+Longitud(***$vectorFamilia_{[i]}$)-1) encuentre valores distintos a “-” (valor de inicialización de la matriz) lo que implica la presencia de al menos un carácter que compone otra palabra.
En cada iteración del ciclo for incrementamos el valor de ***auxFila*** en 1 y decrementamos el valor de ***auxColumna*** en 1 para evaluar la siguiente posición de la matriz en la que queremos colocar la letra.
Finalizado el ciclo for se evalúa el valor de la variable ***contadorCiclos***, si la misma es menor al valor máximo entonces se dimensiona al vector $vectorAux_{[Longitud(palabra)]}$ y mediante el uso de la función ***palabraAVector*** se le asigna el contenido de $vectorFamilia_{[i]}$ se le reasignan a auxFila y auxColumna los valores de ***indiceFila*** e ***indiceColumna*** respectivamente, para luego repetir el ciclo for anteriormente mencionado, esta vez sin evaluar condiciones y asignando a cada posición de la matriz $grilla_{[auxindiceFila,\ auxIndiceColumna]}$ previamente evaluada el contenido del $vectorAux_{[indicePalabra]}$ en donde ***indicePalabra*** inicialmente vale 0 y aumenta en 1 su valor en cada iteración.
La variable de retorno ***error*** se mantiene en 0.
En el caso de que no se cumpla la condición se retorna la variable ***error*** con el valor 1.

invertirPalabra($vectorFamilia_{[i]}$*)*

➢ *Argumento/s:*

Esta función recibe como argumento por valor a la variable $vectorFamilia_{[i]}$.

➢ *Retorno:*

Esta función retorna la variable ***palabraInvertida***.

➢ *Funcionamiento:*

Esta función consiste en un ciclo for que realiza una cantidad de ciclos igual a la Longitud(***vectorFamilia***$_{[i]}$) en el que en cada ciclo el contenido de la variable ***palabraInvertida*** se concatena con el contenido de la Subcadena(palabra, Longitud(***vectorFamilia***$_{[i]}$)-1-i, Longitud(***vectorFamilia***$_{[i]}$)-1-i).
Al ir aumentando el valor de i en 1 en cada ciclo la progresión genera que contenido inicial de de la función Subcadena resulte la última letra de la cadena ***vectorFamilia***$_{[i]}$ y la última iteración resulte en la primera letra de la cadena ***vectorFamilia***$_{[i]}$.
Finalizado el ciclo se retorna la variable ***palabraInvertida***

colocarPalabra(cantidadPalabrasUsadas, familiaDePalabras, grilla, filasGrilla, columnasGrilla)

➢ *Argumento/s:*

Esta función recibe como argumentos por referencia al vector ***familiaDePalabras*** previamente manipulado en la función ***cargarVector*** y a la matriz ***grilla*** previamente inicializada en la función ***inicializarMatriz***.
A su vez recibe como argumentos por valor a las siguientes variables:

- ***cantidadPalabrasUsadas*** (ver referencia).
- ***filasGrilla*** (ver referencia).
- ***columnasGrilla*** (ver referencia).

➢ *Retorno:*

Esta función no tiene variable de retorno.
Sin embargo mediante el pasaje por referencia altera el valor de la matriz (en el caso de que no haya un error por posiciones incorrectas).

➢ *Funcionamiento:*

La función consta de un ciclo do-while en el que se evalúa si hubo algún error al colocar las palabras mediante el valor de la variable ***Reinicio***.
Fuera del ciclo establecemos el valor inicial de reinicio en 0, por lo que al ingresar al mismo entramos a un ciclo for (que va desde 0 hasta la ***cantidadDePalabras***-1) que mediante el uso de la función ***orientarPalabras*** define una ***orientacion*** (valor al azar entre 1 y 8).
Con el valor de la ***orientacion*** ingresamos a un switch que llama a las funciones ***palabraHorizontal***, ***palabraVertical***, ***palabraDiagonalAsc*** y ***palabraDiagonalDesc*** las cuales devuelven el valor 1 en la variable ***Reinicio*** en caso de no poder colocarse la palabra.
Si ***Reinicio*** vale 1 al salir del switch, le asignamos al ***contador*** del ciclo for el valor de salida y al evaluar en la condición while(***Reinicio***=1) el ciclo do-while vuelve a iterar.
Este proceso se repetirá hasta que el total de las palabras (***cantidadDePalabras***) sean colocadas en la grilla de forma correcta.

generarLetraAleatoria(abecedario)

➢ *Argumento/s:*

Esta función recibe como argumento por referencia al vector ***abecedario*** previamente inicializado en la función ***llenarAbecedario***.

➢ *Retorno:*

Esta función retorna la variable ***letraAleatoria***.

➢ *Funcionamiento:*

Asigna a la variable de retorno ***letraAleatoria*** el valor del vector $\textbf{\textit{abecedario}}_{[azar(27)]}$ obteniendo entonces una letra al azar.

llenarRestantes(grilla, filasGrilla, columnasGrilla, abecedario)

➢ *Argumento/s:*

Esta función recibe como argumentos por referencia a la matriz ***grilla*** previamente inicializada en la función ***inicializarMatriz*** y al vector ***abecedario*** previamente inicializado en la función ***llenarAbecedario***.

A su vez recibe como argumentos por valor a las siguientes variables:

- ***filasGrilla*** (ver referencia).
- ***columnasGrilla*** (ver referencia).

➢ *Retorno:*

Esta función no tiene variable de retorno.
Sin embargo al ser un pasaje por referencia modifica de forma directa a la matriz ***grilla***.

➢ *Funcionamiento:*

Consta de dos ciclos for anidados en los que se evalua en el ciclo principal desde ***i***=0 hasta las ***i***=***filasGrilla***-1 y en el ciclo anidado desde ***j***=0 hasta ***j***=***columnasGrilla***-1 si en las posiciones de la $grilla_{[i,j]}$ existe un guión. De ser asi mediante el uso de la funcion ***generarLetraAleatoria*** se le asigna a esa posición una letra al azar del abecedario.
buscarPalabras(familiaDePalabras, cantidadPalabrasUsadas, grilla, filasGrilla, columnasGrilla, vNumeros, cantidadNumeros)

buscarEnSeleccion(grilla, vectorPalabras, palabrasUtilizadas, indicePalabraEncontrada, avanceFilas, avanceColumnas, filaInicial, filaFinal, columnaInicial, columnaFinal)

➢ *Argumento/s:*

➢ *Retorno:*

➢ *Funcionamiento:*

Verificacion(vNumeros Por Referencia, nNumeros Por Valor, posiciones Por Valor, stringFilaColumna Por Valor)

➢ *Argumento/s:*

Esta función recibe por referencia al vector ***vNumeros*** y por valor a las variables ***nNumeros***, ***posiciones*** y ***stringFilaColumna***.

➢ *Retorno:*

Esta función retorna la variable ***posicion***.

➢ *Funcionamiento:*

Al iniciar la función se define la variable ***auxPosicion*** como carácter y se establece un ***contadorCiclos*** en 0.
Dentro de un ciclo do-while definimos la variable ***digitosOK*** con un valor inicial 0.
En el caso de que el contador de ciclos sea mayor a 0, es decir hubo un ingreso erróneo, se le informa al usuario cuales son los valores esperados para su ingreso.
En el caso de que el contador de ciclos sea 0 se le solicita al usuario que ingrese un valor de fila/columna dentro de un rango (el maximo de filas/columnas disponibles).
Se compara mediante dos ciclos for anidados el ingreso del usuario almacenado en ***auxPosicion*** caracter a caracter con el ***vNumeros***, en el caso de existir coincidencias se aumenta en 1 el valor actual de ***digitosOK.***
Pasada esta instancia si la cantidad de ***digitosOK*** es igual a la Longitud(***auxPosicion***) significa que son todos números por lo cual convertimos el valor a número con la función convertirANumero(***auxPosicion***) y le restamos 1 ya que para la matriz el valor 15 está fuera del rango (valores entre 0 y filas/columnas-1). Antes de salir del ciclo do-while se incrementa el valor actual de ***contadorCiclos*** en 1.
El ciclo se repetira mientras que no se cumpla la igualdad entre ***digitosOK*** y Longitud(***auxPosicion***) o posicion<0 o posicion>posiciones(filas o columnas)-1

avances(avanceColumnas Por Referencia, avanceFilas Por Referencia, cambioFilas Por Valor, cambioColumnas Por Valor)

➢ *Argumento/s:*

➢ *Retorno:*

➢ *Funcionamiento:*

buscarPalabras(familiaDePalabras, cantidadPalabrasUsadas, grilla, filasGrilla, columnasGrilla, vNumeros, cantidadNumeros)

➢ *Argumento/s:*

Esta función recibe por referencia a los vectores ***familiaDePalabras*** y ***vNumeros*** y a la matriz ***grilla***.
A su vez recibe por valor a las variables ***cantidadPalabrasUsadas***, ***filasGrilla***, ***columnasGrilla*** y ***cantidadNumeros***.

➢ *Retorno:*

Esta función no tiene variable de retorno. Sin embargo modifica la grilla al encontrar una palabra, cambiándola en su totalidad por guiones.

➢ *Funcionamiento:*

Al iniciar la función, fuera del ciclo do-while, se crea un vector auxiliar ***vectorPalabrasEncontradas***$_{[palabrasUtilizadas]}$ en donde se almacenarán las palabras que el usuario encuentre.
Luego se ingresa en un ciclo do-while que anida a otro ciclo de las mismas características. Previo al ingreso del segundo ciclo se declara un ***contadorCiclos*** con un valor inicial en 0 el cual se utiliza dentro del segundo ciclo para diferenciar los mensajes de error.
El segundo ciclo inicia con un clear de pantalla y mostrando la matriz ***grilla*** (previamente manipulada en la función ***llenarRestantes***), debajo de la grilla exhibe, mediante el uso de un ciclo for que recorre al ***vectorPalabras*** comparando su contenido con el de ***vectorPalabrasEncontradas*** mediante un condicional, e imprime en pantalla las palabras que no coinciden.
En el caso de que no sea la primera repetición del ciclo existe, a continuación, un condicional con mensajes de error (no se realizó una selección de más de una celda o el desplazamiento en diagonal no es a 45°).
Por último en este ciclo anidado se le asigna mediante el uso de la función ***Verificacion*** (que se encarga de que los valores ingresados sean válidos para la ***grilla***) valores a las variables ***filaInicial***, ***columnaInicial***, ***filaFinal*** y

columnaFinal.
Con dichos valores calculamos el ***cambioFilas*** y ***cambioColumnas*** como ***filaFinal-filaInicial*** y ***columnaInicial-columnaFinal*** respectivamente. Previo a la finalización del ciclo se incrementa en 1 el valor del contador de ciclos.
El ciclo se repetirá mientras el cambio de filas y columnas sea 0 o mientras los módulos de ambos sean distintos de cero y entre sí. Es decir hasta que los valores de ***cambioFilas*** y ***cambioColumnas*** sean válidos.
Una vez fuera del ciclo anidado mediante el uso de la función avances establecemos el valor de las variables ***avanceColumnas*** y ***avanceFilas***.
Tambien inicializamos la variable ***cantidadPalabrasEncontradas*** en 0, el ***indicePalabraEncontrada*** en -1 y la ***palabraEncontradaEnEstaRonda*** en 0 como condiciones iniciales de control.
Mediante el uso de la función ***buscarEnSeleccion*** modificamos (o no) el valor de ***palabraEncontradaEnEstaRonda*** (si se encuentra una palabra la función retorna 1 y le asigna por referencia al ***indicePalabraEncontrada*** la posición del vector ***familiaDePalabras*** en la que fue encontrada).
Con un condicional entonces si el valor de ***palabraEncontradaEnEstaRonda*** y el valor de ***indicePalabraEncontrada*** se mantuvieron en su estado inicial (0 y -1) se comunica que no se encontraron palabras en la selección, caso contrario se le asigna al ***vectorPalabrasEncontradas***$_{[indicePalabraEncontrada]}$ el valor de ***familiaDePalabras*** $_{[indicePalabraEncontrada]}$ y se muestra por pantalla una felicitación indicando cual es la palabra que se encontró.
Además mediante un ciclo for que recorre el vector de palabras encontradas desde i=o hasta ***palabrasUtilizadas***-1 si ***familiaDePalabras*** $_{[i]}$***=vectorPalabrasEncontradas*** $_{[i]}$ se suma 1 al valor actual de ***cantidadPalabrasEncontradas.***
Mientras ***cantidadPalabrasEncontradas!=palabrasUtilizadas*** se le consulta al usuario si desea seguir buscando palabras o prefiere rendirse. El ciclo do-while principal se repetirá mientras que el usuario no ingrese la frase “me rindo” en esta instancia y a también mientras que las palabras encontradas no sean iguales a las palabras utilizadas.
Finalizado el ciclo principal por cualquiera de los 2 motivos, se analiza la condicion de igualdad entre las variables ***cantidadPalabrasEncontradas*** y ***palabrasUtilizadas*** y de ser verdad se imprime un mensaje explicando que ganaste, caso contrario se muestra un mensaje explicando que te rendiste.

nombreFuncion

➢ *Argumento/s:*

➢ *Retorno:*

➢ *Funcionamiento:*

```
variable ← asignacion
Dimensionar VectorMatriz[tamaño]
```