TEC | Tecnológico de Costa Rica

IC2001 ESTRUCTURAS DE DATOS

GRUPO 2

JOSE PABLO AGUERO MORA 2021126372

KATERINE GUZMÁN FLORES 2019390523

MANUAL TÉCNICO PROYECTO MATRIX

I SEMESTRE 2022

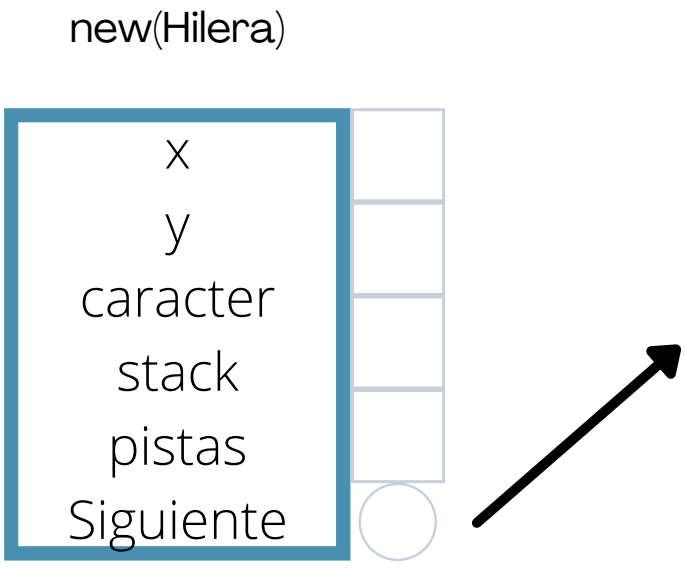
ESTRUCTURA DE TIPO HILERA

Se crea un struct de tipo hilera que funciona como estructura de datos donde se van a albergar los grupos de caracteres en una lista enlazada.
Esta estructura posee seis campos los cuales son: posición en x, posición en y con respecto a la pantalla, el stack de tipo char donde se albergan los caracteres de cada hilera, una variable de tipo entero para llevar el conteo de las pistas, además de dos variables de tipo entero para llevar la cantidad de letras generadas y agrupaciones por pistas para las estadísticas finales de la ejecución.

CÓDIGO

```
// Se usa un struct de tipo hilera
typedef struct Hilera {
    int X;
    int Y;
    char caracter;
    char stack[12]; // arreglo de caracteres que van cayendo
    int pistas = 1;
    Hilera* Siguiente; // Puntero que apunta a la siguiente hilera
}*Ptr_Hilera;
```

DIBUJO



INICIALIZAR HILERA

Se inicializa la hilera con un puntero que apunta a Null, es decir se le da un valor neutro..

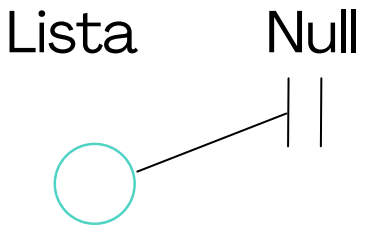
CÓDIGO

```
Ptr_Hilera Agrupaciones;
Ptr_Hilera Nuevo;

// Se inicializan las hileras de la lista enlazada
InicializarHilera(Agrupaciones);
```

```
// Recibe la lista de hileras como parametro por referencia que tiene la forma tipo Ptr_Hilera
void InicializarHilera(Ptr_Hilera& Lista)
{
    Lista = NULL;
}
```

DIBUJO

Agrupaciones = Lista
Lista = Null



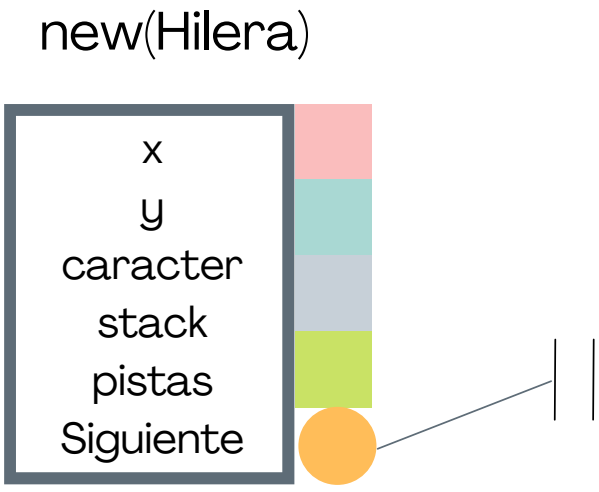
CREAR AGRUPACIÓN

Cuando se ejecuta esta función se crea una nueva variable anónima de tipo hilera. Posteriormente, en un primer recorrido se le asigna la posición 20 en x y en y, que sería la hilera que se pinta en pantalla. Aquí mismo se llama a otro función para generar un caracter aleatorio que esté dentro de las letras que conforman el abecedario. Finalmente se le asigna un puntero a NULL, para poder seguir creando la lista enlazada.

CÓDIGO

```
// Se crea una hilera
Ptr_Hilera CrearAgrupacion(int cod)
{
    Ptr_Hilera Agrupacion = new(Hilera);

    xInicial += 20; // Se va desplazando en x cada 20 pixeles
    Agrupacion->X = xInicial;
    Agrupacion->Y = yInicial; // siempre se inician en esta posicion porque es el inicio de la pantalla
    Agrupacion->caracter = GenerarRandom(); // se le asigna caracter aleatorio
    Agrupacion->Siguiente = NULL;
    return Agrupacion; // Devuelve la agrupacion creada
}
```

DIBUJO



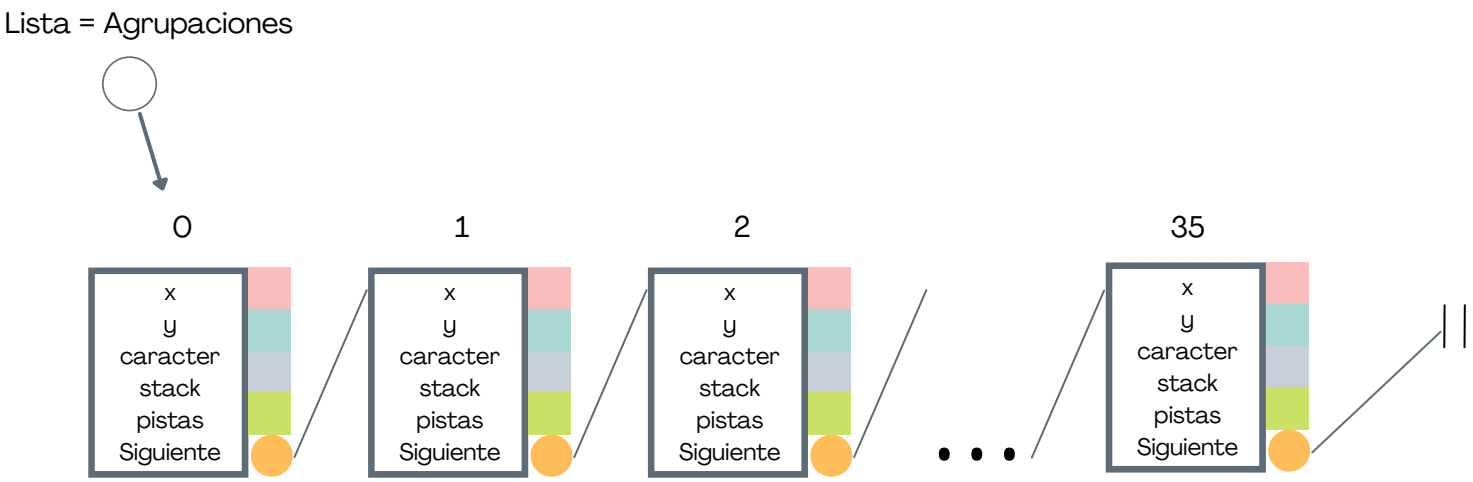
AGREGAR FINAL

Al listar los nodos, estos se van acomodando de izquierda a derecha como se muestra en la imagen. Los nodos se van agregando al final de la lista y el puntero lista pasa a ser el segundo en la lista y así sucesivamente, los va acomodando en orden ascendente hasta llegar a las 35 hileras que es el límite establecido.

CÓDIGO

```
// Se van agregando las hileras a la lista
void AgregarFinal(Ptr_Hilera& Lista, Ptr_Hilera& Nuevo)
{
    Ptr_Hilera Aux;
    Aux = Lista;
    if (Aux != NULL) // Si hay al emnos un elemento
    {
        while (Aux->Siguiente != NULL) // si es el ultimo nodo
        {
            Aux = Aux->Siguiente; // Se le asigna el siguiente nodo
        }
        Aux->Siguiente = Nuevo;  // si ya llegamos a null llegamos al ultimo elemento y se engancha al final
    }
    else
    {
        Lista = Nuevo; // Se agrega nuevo al final
    }
}
```

DIBUJO



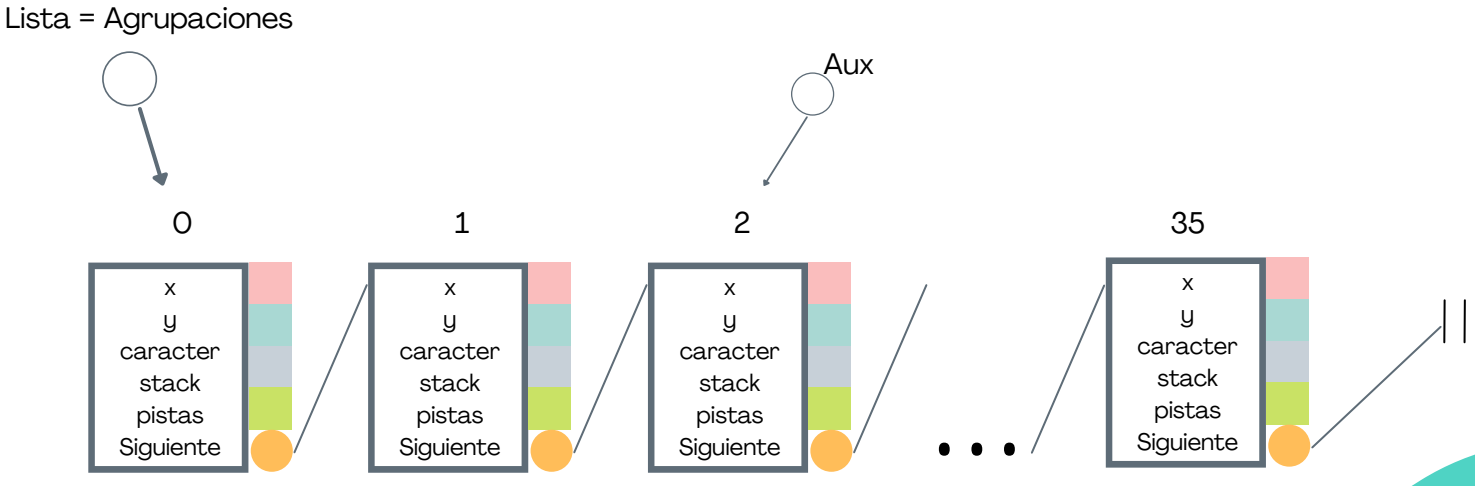
LISTAR PISTAS

Esta función, recibe la lista de agrupaciones, se genera un contador auxiliar para ir contando las pistas. Además se crea una variable de tipo puntero hilera llamada auxiliar para ir recorriendo toda la lista y obtener los datos requeridos y una variable de tipo string llamada oración para ir concatenando la información que se ocupa guardar en el archivo.. En el ciclo cuando ya se recorre la lista, se concatena la palabra lista con el contador para saber cual número de pista es, además que se obtiene del struct la cantidad de hileras que contiene cada pista y esta información se envía al archivo editable.

CÓDIGO

```
void ListarPistas(Ptr_Hilera& Lista)
{
    int Contador = 1;
    Ptr_Hilera Aux;
    Aux = Lista;
    string oracion; // Se guarda en esta variable el texto que se va guardando en el artículo
    while (Aux != NULL)
    {
        oracion = "Pista " + to_string(Contador); // Indica el numero de la pista
        oracion += " = Cantidad de agrupaciones: " + to_string(Aux->pistas); // Indica la cant
        oracion += "\n";
        EditarArchivo(oracion); // Se incluye la información anterior al archivo

        // Se imprime la misma informacoón en pantalla
        printf("Pista %d ", Contador);
        printf("= Cantidad de agrupaciones: %d ", Aux->pistas);
        cout << endl;
        Aux = Aux->Siguiente;
        Contador++;
    }
}
```

DIBUJO



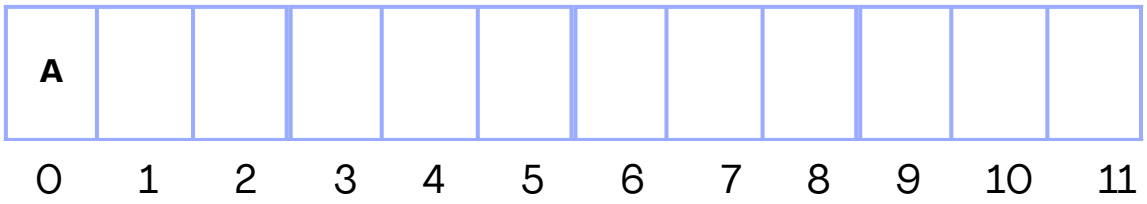
DESPLAZAR

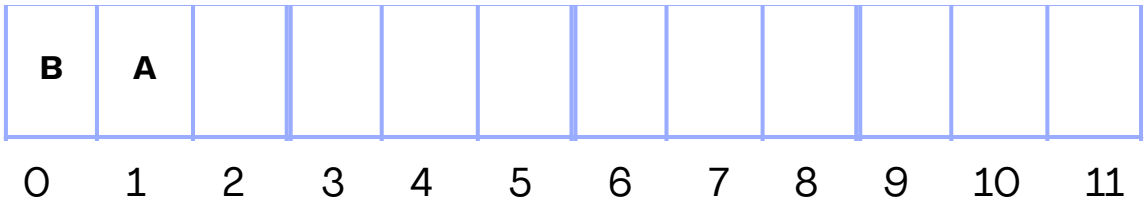
La función desplazar lo que permite es ir moviendo los caracteres aleatorios dentro de la pila de stack de caracteres hacia la izquierda para que se pueda ingresar un nuevo caracter de forma inifinita. De esta forma sale un caracter por la derecha para que se dibuje y entra un nuevo caracter próximo a ser dibujado.

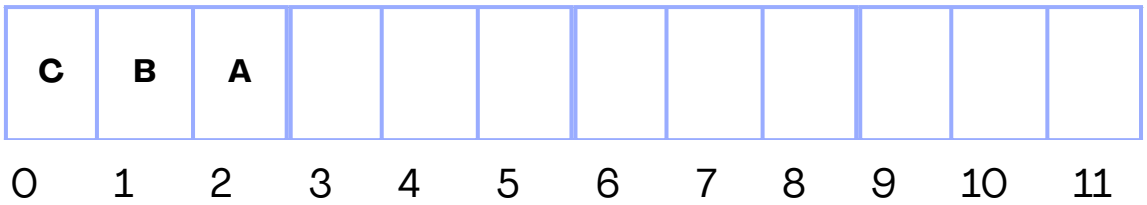
CÓDIGO

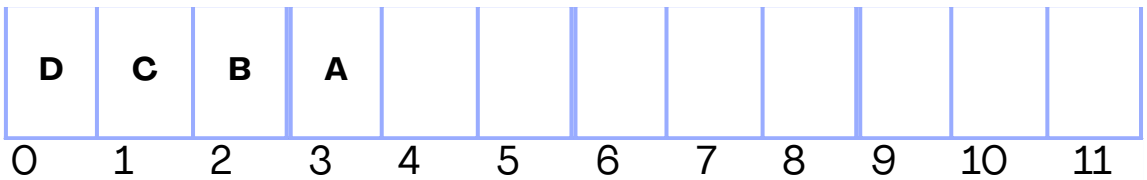
```
void desplazar(char stack[12]) {
    stack[11] = stack[10];
    stack[10] = stack[9];
    stack[9] = stack[8];
    stack[8] = stack[7];
    stack[7] = stack[6];
    stack[6] = stack[5];
    stack[5] = stack[4];
    stack[4] = stack[3];
    stack[3] = stack[2];
    stack[2] = stack[1];
    stack[1] = stack[0];
}
```

EJEMPLO AL INGRESAR ABCD AL STACK







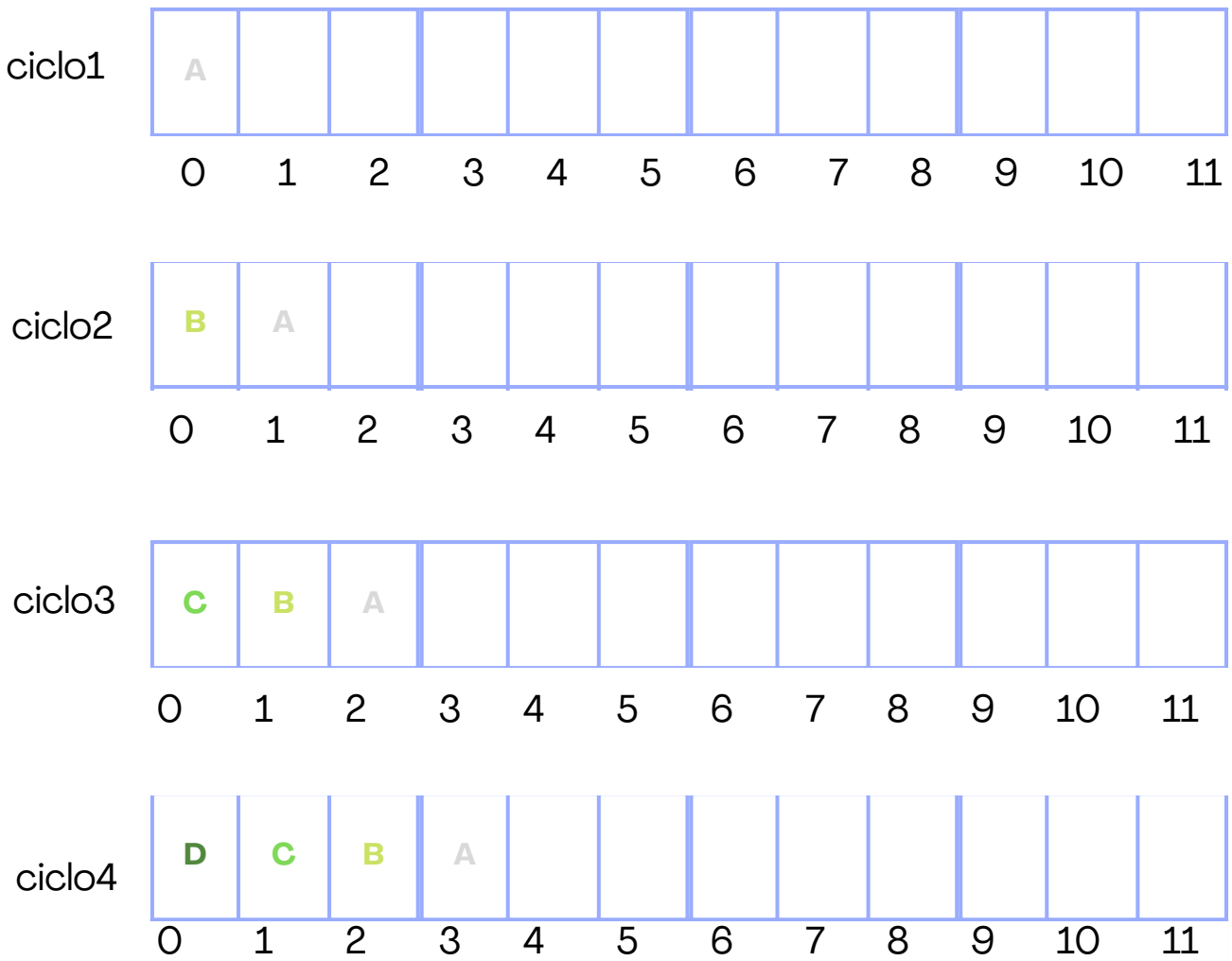


DIBUJAR

Esta es la función principal del programa y lo que es lo siguiente:

- Primero se declara una variable global a utilizar llamada actual donde se va a albergar los caracteres que vienen del stack, uno a uno.
- Actual va a tomar la posición 0 del stack en un primer ciclo y lo va a pintar en pantalla con un color respectivo mediante una función de allegro. Este primer caracter irá de color blanco y los demás que serían espacios en blanco los pintaría en diferentes tonos de verde.
- En un segundo ciclo este primer elemento pasa a la derecha y se ingresa el nuevo elemento del stack a la izquierda.
- Dibujar imprime ese nuevo elemento en un color verde claro.
- De esta forma se van corriendo los caracteres dentro del stack por la función desplazar, la variable actual dentro de dibujar la toma y la pinta de diferentes tonos de verdes, entonces se ve como si fueran en degradación hasta que llega a pintarse un caracter negro en pantalla.
- A modo que se van desplazando los caracteres a la derecha, se van generando nuevos caracteres aleatorios ya que esto se hace en un ciclo infinito dentro del main.

EJEMPLO AL DIBUJAR ABCD EN PANTALLA



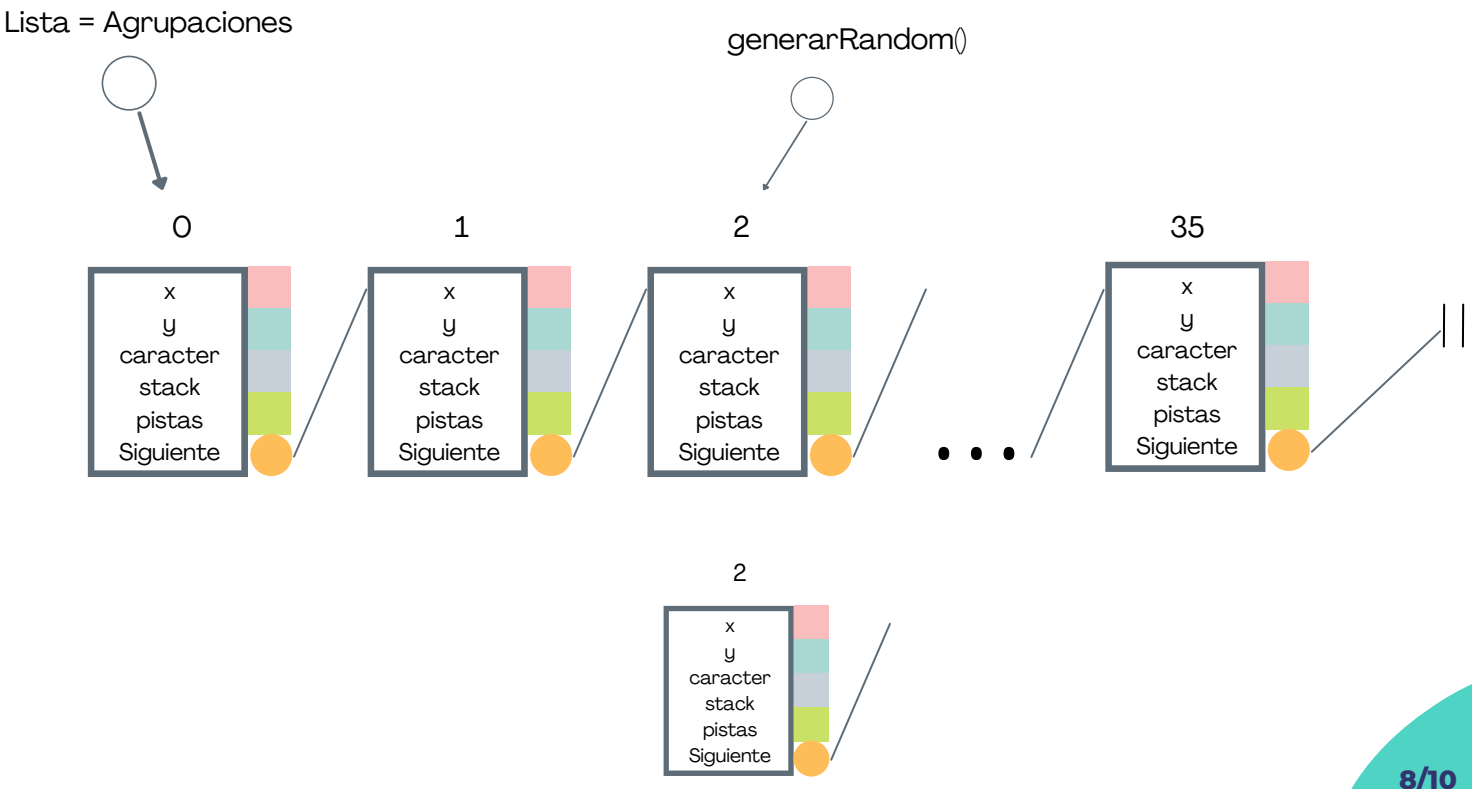
AGRUPACIÓN ALEATORIA

Esta función lo que hace es recorrer toda la lista y llevar un contador para contar las hileras totales. Se utiliza un llamado a otra función auxiliar que permite generar un numero aleatorio según el total de estas hileras y luego mediante un ciclo while se recorre toda la lista y se detiene cuando se encuentra la hilera con este índice en la lista enlazada. Una vez encontrado se devuelve la hilera. Esta función se utiliza dentro del main para generar una hilera en una posición aleatoria en pantalla.

CÓDIGO

```
Ptr_Hilera AgruAleatoria(Ptr_Hilera& Lista)
{
    // se recorre la lista para saber cuantas hileras en total hay
    int contador = 0;
    Ptr_Hilera Aux;
    Aux = Lista;
    while (Aux != NULL)
    {
        contador += 1;
        Aux = Aux->Siguiente;
    }

    // Me devuelve la hilera con el numero ramdon
    int ra = crearRandom(contador);
    Ptr_Hilera Aux2;
    Aux2 = Lista;
    int cont2 = 1;
    while (cont2 < ra)
    {
        cont2 += 1;
        Aux2 = Aux2->Siguiente;
    }
    return Aux2;
}
```

DIBUJO



CREAR/ EDITAR ARCHIVO

La función crear archivo permite crear un archivo de escritura para poder guardar la información de las estadísticas en un archivo txt cuando finalice el programa y visualizar los resultados. Si el archivo es NULL se imprime que hubo un problema al crearlo, de lo contrario lo crea y se cierra.

CÓDIGO

```
void CrearArchivo()
{
    FILE* archivo;
    fopen_s(&archivo, "stats.txt", "w"); //crea el archivo de forma escritura

    if (NULL == archivo) { // si el archivo es NULL entonces hubo problemas al
        fprintf(stderr, "No se pudo crear archivo %s.\n", "resultados.txt");
        exit(-1);
    }
    fclose(archivo); //cierra el archivo.

}

// programa principal
```

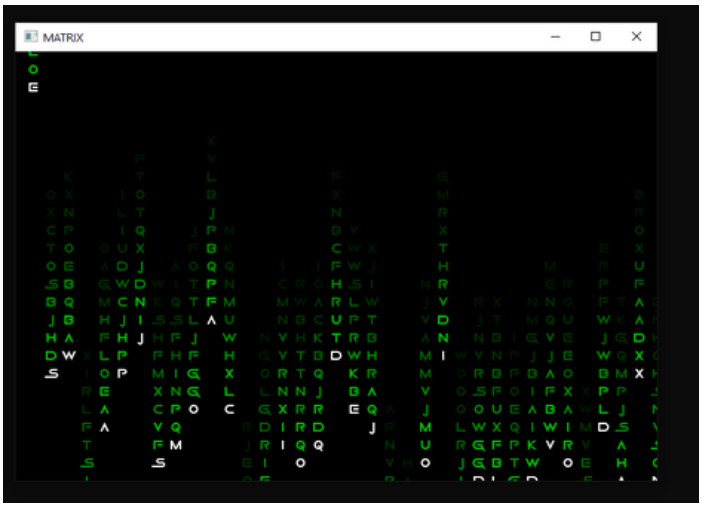
EDITAR ARCHIVO

Editar archivo funciona como crear archivo, sin embargo, el archivo que se abre será el creado anteriormente y se le van ingresando los datos de las estadísticas desde donde es llamada que sería la función listarPistas.

```
void EditarArchivo(string cod)
{
    FILE* archivo;
    fopen_s(&archivo, "stats.txt", "a"); // crea el archivo de forma que se pueda editar

    if (NULL == archivo) { // si el archivo es NULL entonces hubo problemas al crearlo ge
        fprintf(stderr, "No se pudo crear archivo %s.\n", "resultados.txt");
        exit(-1);
    }
    else {
        //Si se pude crear el archivo, entonces:
        fprintf(archivo, "%s\n", cod);
    }
    fclose(archivo); //cierra el archivo.

}

void ListarPistas(Ptr Hilera& Lista)
```

EJECUCIÓN DEL PROGRAMA Y PRUEBAS

Al ejecutar el programa sale esta pantalla con un sonido previamente establecido. Se puede observar como van cayendo las hileras desde el inicio de la pantalla, como van cambiando el color de los carateres aleatorios, además de el cambio de posición de las hileras pintadas.



IMPRESIÓN EN PANTALLA DE RESULTADOS Y ARCHIVO CREADO