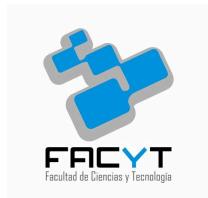




UNIVERSIDAD DE CARABOBO
FACULTAD DE CIENCIA Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN



Proyecto: Emisora de Radio

Profesor:
Gabriel Peraza
Sección 1

Estudiante:
Gabriel Becerra
C.I. 31.654.243

17 de Octubre de 2025

Descripción del problema

Explicación general del enunciado y objetivos:

Debido al ineficiente trabajo manual de sus antiguos encargados, la emisora “Éxitos FM” nos ha contratado para automatizar el proceso de creación de sus horarios semanales.

Algunos de nuestros objetivos más importantes para esta tarea son:

- Garantizar que se suplan los horarios estelares con los shows y canciones de más alto rating disponibles.
- Asegurar que se cumpla el número mínimo de reproducción diaria de las cuñas publicitarias.
- Respetar el límite de reproducción de las canciones (no más de una vez cada 4 hrs) y el límite de shows radiales (1 por día).
- Respetar la regla de separación entre elementos y garantizar que no haya más de un segundo en el que no se esté reproduciendo nada.

Contexto y propósito del software desarrollado:

Desarrollaremos un programa que, dado una serie de archivos de entrada ([canciones.in](#), [shows.in](#), [publicidad.in](#)) que contengan los eventos a nuestra disposición, su duración, calificaciones y/o números de reproducción mínima, genere para cada dia de la semana, de lunes a domingo, un horario de programación radial que cumpla las limitaciones estipuladas, ajustado de manera inteligente para incluir shows y canciones en horarios óptimos según su calificación, con la finalidad de mejorar el rating de la emisora.

Análisis de la solución

Cómo se organizan y almacenan los datos:

Ya que disponemos de tres tipos de entrada diferente, uno por cada archivo, se hace uso de una serie de arreglos unidimensionales del tipo de dato “canción”, “show”, “publicidad” con el tamaño máximo permitido por cada uno según nuestro enunciado y un arreglo extra del tipo de dato “evento” para control interno de nuestro programa. Con respecto a las funciones de lectura, en las tres se utiliza la función “fscanf()” con el formato de lectura “[^\t]” que lee hasta encontrarse con el primer tabulador de la línea, esto para leer el nombre del evento, luego sus demás datos y almacenarlos en el campo de registro correspondientes; con la diferencia de que en la función de leer publicidad, se lee hasta final de archivo, ya que este no contiene un indicio en la primera línea de la cantidad de elementos a leer.

```
main

//Arreglos de todos los eventos disponibles y de 'eventos' para llevar registro de la
programacion
    cancion canciones[1000];
    publicidad cuñas[100];
    show shows[15];
    evento eventos[10000];

    int nroCan = 0, nroShow = 0, nroPub = 0, nroEvent = 0;

    //Leemos entradas
    leerCanciones(canciones,&nroCan);
    leerPublicidad(cuñas,&nroPub);
    leerShows(shows,&nroShow);
```

Figura 1. Fragmento de código en main donde se declaran todos los arreglos y se llama a las funciones de lectura.

Se mantuvieron las funciones de lectura separadas por tipo de archivo para favorecer la modularidad, lo que facilita detectar errores en la entrada y simplifica futuras modificaciones.

```

    leerCanciones()

    fscanf(in, "%d", nroCanciones);
    fgetc(in); //Consumimos el salto de Linea

    for(int i=0;i<(*nroCanciones);i++){
        //Variables auxiliares
        char nombreTemp[100];
        int min,sec,punt;

        if(fscanf(in,"%[^t]\t%d\t%d\t%d\n",nombreTemp,&min,&sec,&punt) != 4){ //Leemos nombre
            hasta el primer \t
            printf("Error de lectura en la linea %d de canciones.in\n",i+1);
            continue;
        }

        strcpy(canciones[i].nombre,nombreTemp);
        canciones[i].min = min;
        canciones[i].seg = sec;
        canciones[i].punt = punt;

        canciones[i].ultimaHora = -9999; // La inicializamos como no usada

    }
}

```

Figura 2. Ciclo de lectura de la función leerCanciones (archivo de entrada previamente validado).

```

    leerPublicidad()

    char nombreTemp[100];
    int segs, veces;
    *nroCuñas = 0;

    while(fscanf(in, "%[^t]\t%d\t%d\n",nombreTemp,&segs,&veces) == 3){

        strcpy(cuñas[*nroCuñas].empresa, nombreTemp);
        cuñas[*nroCuñas].duracionSeg = segs;
        cuñas[*nroCuñas].repeticiones = veces;
        cuñas[*nroCuñas].repActual = 0;
        (*nroCuñas)++;
    }
}

```

Figura 3. Flujo de la función leerPublicidad.

Justificación de las decisiones tomadas:

Al diseñar el programa se priorizó la modularidad y la legibilidad, para facilitar depuración y evaluación. Cada grupo de tareas (lectura de archivos, validación de restricciones, generación de programación, y consultas) se encapsuló en funciones independientes, cosa que a lo largo del desarrollo permitió solucionar problemas de manera aislada sin tener que re-trabajar todo.

Para la selección de canciones y publicidades, se implementó un método aleatorio ponderado basado en su nivel de popularidad o prioridad. Con lo que se buscaba garantizar diversidad y evitar patrones de repetición en la programación diaria, y aunque no está garantizado, se mantiene la alta probabilidad de que los contenidos más valorados aparezcan con mayor frecuencia. Este enfoque intenta emular el realismo de cómo se comportaría una emisora de la vida con la variabilidad y restricciones impuestas en el enunciado.

Las variables de control como ultimaHora, repActual y usadoHoy establecidas dentro de las estructuras tienen la finalidad de facilitar la validación de restricciones en tiempo de ejecución (repetición de canciones cada 4 horas, shows únicos por día y repeticiones de cuñas). Se evitan cálculos externos y se mantiene la coherencia de los datos.

Otra función central en la estructura del programa es insertarShowsDistribuidos(), cuyo propósito es completar los bloques horarios entre las franjas estelares mediante la inserción de shows secundarios, canciones y publicidades. Esta función actúa como un módulo de distribución temporal inteligente, evaluando la duración restante del bloque y seleccionando dinámicamente qué tipo de evento ubicar en cada momento, siempre respetando restricciones. Además continúa usando el mismo sistema de selección aleatoria ponderada por popularidad o preferencia, para variedad y realismo.

Finalmente, el programa centraliza la planificación en la acción ProgramarDia(), que coordina todas las anteriores funciones auxiliares. Actúa como punto de control principal de todo el algoritmo, representa el flujo general del día y garantiza que todos los componentes interactúen de forma sincronizada.

Estructuras de datos

```
global

typedef struct {
    char nombre[51];
    int min;
    int seg;
    int punt;           // Popularidad de la cancion del 1 al 100
    int ultimaHora;    // Para controlar repeticiones
} cancion;

typedef struct {
    char empresa[31];
    int duracionSeg;
    int repeticiones;
    int repActual;     // Para control interno
} publicidad;

typedef struct {
    char nombre[101];
    int durMin;
    int durSeg;
    int segmentos;
    int preferencia;  // Rating del show del 1 al 10
    int usadoHoy;     // Para no repetir mas de una vez por dia
} show;

typedef struct {
    int tiempo;
    char tipo;         // 'S' show, 'C' cancion, 'P' publicidad
    int dia;           //1-Lunes, 2-Martes...
    char nombre[120];
} evento;           //Para consulta de usuario al final del programa
```

Figura 4. Declaración de todas las estructuras de datos utilizadas.

Cómo es posible observar, aparte de la información proporcionada por el archivo, también se implementaron en las estructuras campos netamente utilizados para validación (Ej. usadoHoy para evitar repetir un show mas de una vez al dia, repActual para llevar el control de cuantas veces se ha repetido una cuña y ultimaHora para respetar la restricción de canciones cada 4 horas).

También se implementó la estructura “eventos” como contenedor unificado de toda la programación semanal que es escrita en los archivos de salida, esto para que todas las funciones de consulta puedan trabajar con una sola lista ordenada.

Además de los arreglos de estructura principales, se utilizan varias veces a lo largo del programa arreglos unidimensionales y bidimensionales, algunas instancias son:

```
ProgramarDia()

// Elegir shows estelares
int estelares[3];
elegirShowsEstelares(shows, nroShows, estelares);
for(int i = 0; i < 3; i++){
    if(estelares[i] >= 0) shows[estelares[i]].usadoHoy = 1; //Marcarlos como usados
}
```

Figura 5. Elección de los shows que protagonizan las franjas estelares en el dia.

```
ProgramarDia()

// Bloques fijos de estelares
int franjas[3][2] = {
    {7*3600, 9*3600},
    {12*3600, 14*3600},
    {18*3600, 19*3600}
};
```

Figura 6. Mapeo de los horarios estelares mediante una matriz.

```
elegirCancionparaSeparador()

//Sorteo ponderado
int r = rand() % total;
for (int i=0; i<nroCan; i++) {
    if (r < pesos[i]) return i;
    r -= pesos[i];
}
return -1;
```

Figura 7. Sorteos ponderados presentes en varias instancias en las que hay que elegir canciones o publicidades para llenar huecos.

El uso de sorteos ponderados se eligió para mantener la aleatoriedad en la programación, pero respetando la prioridad del rating; de esta forma se garantiza variedad sin perder la lógica de popularidad.

Algoritmos implementados

El uso de algoritmos estructurados está mayormente presente en las dos funciones más trabajadas del programa, estas son ProgramarDia(), que sería la función ‘principal’ de nuestro programa, con llamadas a todas las otras funciones auxiliares, y una llamada en main por cada día de la semana; y la función insertarShowsDistribuidos(), debido a que se le dio una especial importancia al flujo de insercion de shows a lo largo del día, a diferencia de las funciones de insertar publicidades o canciones, que se rigen de una serie de sorteos ponderados según restricciones y reglas a cumplir, la función de inserción de shows utiliza un algoritmo un poco más elaborado para garantizar una inserción uniforme, esparcida y ‘aleatoriamente ponderada’ por así decirlo.

Figura 8. Algoritmo simplificado de la función insertarShowsDistribuidos en lenguaje pseudoformal.

```
function insertarShowsDistribuidos(tiempoAct,tiempoFin,dia,canciones,shows,publicidad): int
    obtenerShowsNormales
    if(noHayshowsDisponibles)then
        while(noHaTerminadoELDia)do
            intentarCancion0publi
            if(noCupoNada)then
                //Avanzar para evitar bucles inf.
            endif
        endwhile
        return tiempoAct
    endif
    //Caso normal donde si tenemos shows
    while(noHaTerminadoELDia)do
        if(sePuedenInsertarShows)then
            if(noCabeNinguno)then
                reintentarEn30min
            endif
            seleccionarCandidato
            for segmento <- 1 to nroSegmentos ^ noHaTerminadoELDia do
                if(noQuedaTiempo)then
                    break
                endif
                escribirShow
                if(quedanSegmentosyTiempo)then
                    insertarPublicoCancion
                endif
            else
                rellenarEntreShows
            endif
        return tiempoAct
    endfunc
```

La función trabaja manejando el tiempo en segundos del día, así cada que inserta un evento, suma la duración al tiempo actual y se lo retorna a ProgramarDia(). También está blindada para lidiar con una multitud de casos (escasez de canciones, de shows, de

publicidades) Para garantizar que incluso en los escenarios más poco realistas, el horario semanal sea llenado con éxito.

Figura 9. Procedimiento de programación diaria simplificado en lenguaje pseudoformal.

```
procedure ProgramarDia(dia,canciones,shows,publicidad)
    nombrarArchivoSalida
    reiniciarVariables
    elegirShowsEstelares
    for franja <- 1 to 3 do
        programarShowEstelar
        RellenarFindeFranja
    endfor
    RellenarHastaMedianoche
endproc
```

A pesar de ser un algoritmo bastante simple a primera vista, esta función contiene las llamadas a todas las funciones auxiliares anteriores, cada ‘rellenarFindeFranja’ es en realidad una llamada a insertarShowsDistribuidos(), donde se hacen cosas como sortear las canciones según su rating, las publicidades según sus necesidades de reproducción, validar restricciones cada vez que vaya a imprimirse algo, etc, por lo que podría decirse que la función ProgramarDia(), ayudada del extenso algoritmo presente en insertarShowDistribuido(), es la función “Principal” del programa, utiliza un enfoque basado en el llenado de los horarios estelares, partiendo de la idea de que nuestra prioridad principal es suplir esas franjas horarias, y de la garantía de que siempre tendremos un show disponible para cada una de esas horas.

Pruebas realizadas

A lo largo del desarrollo del software fue necesario ir haciendo corridas de prueba para verificar que todo estuviera funcionando en orden, el comportamiento más interesante fue observado en los siguientes casos

-Pocos shows, pocas canciones, pocas publicidades (Peor Caso)

Con los siguientes archivos de entrada:

```
//shows.in  
5 Nvidia GeForce 15 8  
Range Time with Franklin Cooper 10 15 5 7  
El misterio de CROW Sesenta-y-Cuatro 16 27 4 9  
The Escapists Audio Walkthrough Full Series 45 58 3 4  
MumboJumbo Redstone Tutorials 5 25 6 7  
A comprehensive Analysis of Robloxian Anatomy 15 23 4 7
```

```
//canciones.in  
6  
Brave Heart 4 12 90  
Butter-fly 5 17 81  
Break the Chain 5 9 77  
Diver 4 10 91  
Black Night Town 1 39 69  
Zetsu Zetsu 1 44 70
```

```
//publicidad.in  
Nvidia GeForce 15 8  
Nintendo 30 4  
Movistar 20 10
```

3 publicidades, 6 canciones y 5 shows (3 estelares, 2 normales).

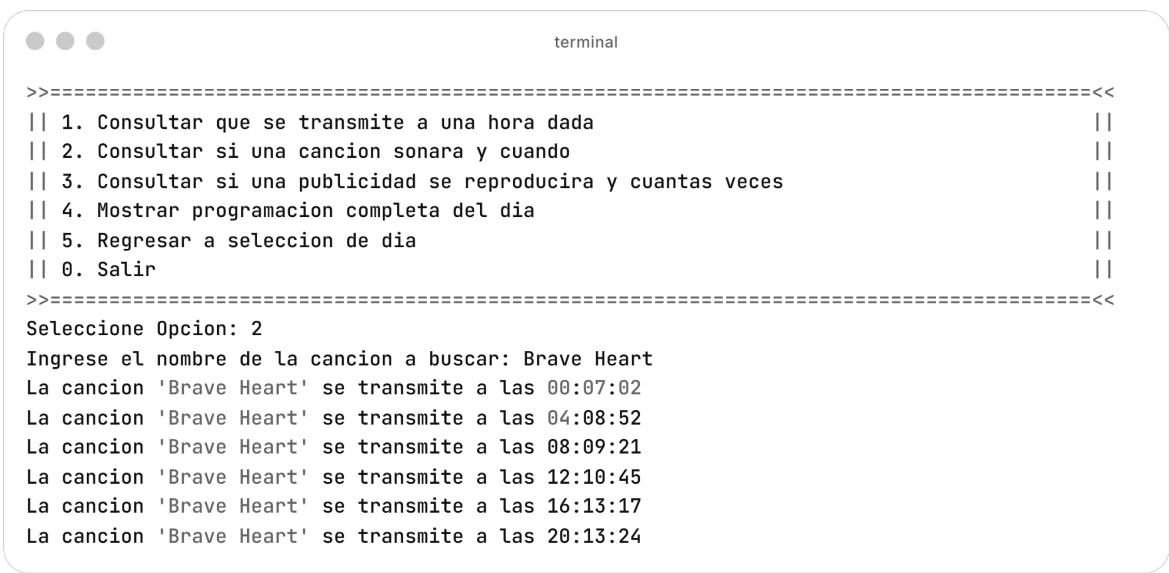
Este es un caso obviamente, excesivamente irreal, pero es uno que técnicamente entra dentro de los parámetros para los que debe estar preparado nuestro programa.

Una cantidad tan minúscula de eventos que además, están limitados a una cierta cantidad de reproducciones obviamente no puede cubrir un dia completo, por lo que en este tipo de casos el programa debe activar su “salida de emergencia” e irse por lo más seguro

```
file  
//Se observa un inicio del dia saturado de publicidad, con canciones ocasionales donde entran en hueco  
00:05:00 P Nvidia GeForce  
00:05:16 P Movistar  
00:05:37 P Nvidia GeForce  
00:05:53 P Nvidia GeForce  
00:06:09 P Nvidia GeForce  
00:06:25 P Nvidia GeForce  
00:06:41 P Movistar  
00:07:02 C Brave Heart  
00:11:15 P Nvidia GeForce  
00:11:31 P Nintendo  
00:12:02 P Nintendo  
00:12:33 P Movistar  
00:12:54 P Movistar  
00:13:15 P Movistar  
00:13:36 C Break the Chain  
00:18:46 C Zetsu Zetsu  
00:20:31 P Movistar  
00:20:52 C Black Night Town  
00:22:32 P Movistar  
00:22:53 P Nintendo  
00:23:24 P Nvidia GeForce  
00:23:40 P Nintendo  
00:24:11 C Diver  
00:28:22 P Nvidia GeForce  
00:28:38 P Nvidia GeForce  
00:28:54 P Movistar  
00:29:15 P Nvidia GeForce  
00:29:31 C Butter-fly  
00:34:49 P Nintendo  
00:35:20 P Nintendo  
00:35:51 P Nintendo  
00:36:22 P Nintendo  
00:36:53 P Movistar  
-----
```

Figura 13. Comportamiento del archivo de salida en casos de escasas entradas.

Como las publicidades son el único evento que no tiene restricción de reproducción diaria y prácticamente pueden reproducirse ilimitadas veces son la única opción que le queda al programa en un caso como este. Los pocos shows que están afuera de los horarios estelares hallan también su hueco en la programación del día, mientras que las canciones al ser tan pocas, aparentan tener una forma de patrón, ya que se reproducen al inicio cuando están disponibles, y luego quedan esperando su próxima reproducción luego de 4 horas, así hasta el final del día.



```

terminal
>>=====
|| 1. Consultar que se transmite a una hora dada      ||
|| 2. Consultar si una cancion sonara y cuando        ||
|| 3. Consultar si una publicidad se reproducira y cuantas veces ||
|| 4. Mostrar programacion completa del dia          ||
|| 5. Regresar a seleccion de dia                   ||
|| 0. Salir                                         ||
>>=====

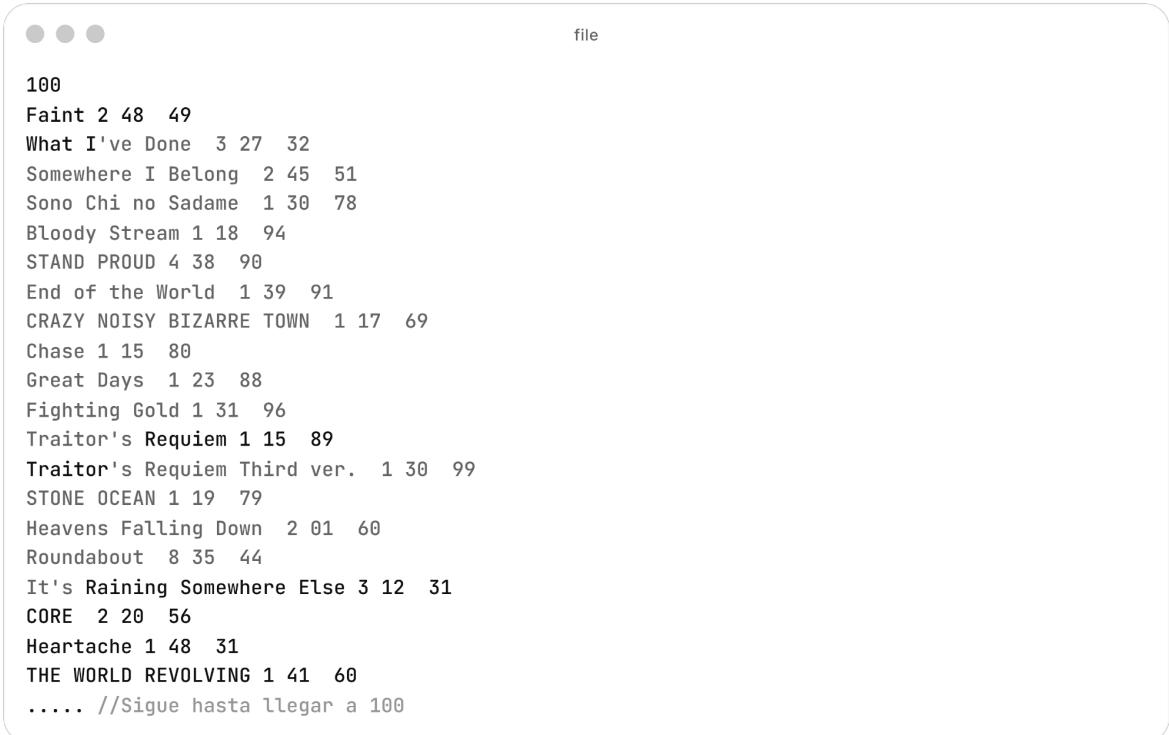
Seleccione Opcion: 2
Ingrese el nombre de la cancion a buscar: Brave Heart
La cancion 'Brave Heart' se transmite a las 00:07:02
La cancion 'Brave Heart' se transmite a las 04:08:52
La cancion 'Brave Heart' se transmite a las 08:09:21
La cancion 'Brave Heart' se transmite a las 12:10:45
La cancion 'Brave Heart' se transmite a las 16:13:17
La cancion 'Brave Heart' se transmite a las 20:13:24

```

Figura 14. Observación del comportamiento de canciones mediante el menú de consulta por terminal.

-Pocos shows, pocas publicidades, muchas canciones

Para tratar de remediar un poco lo desgradable que se escucharía una estación de radio con puras publicidades durante todo el día, podemos observar lo que ocurre cuando la saturamos de canciones.



```

file
100
Faint 2 48 49
What I've Done 3 27 32
Somewhere I Belong 2 45 51
Sono Chi no Sadame 1 30 78
Bloody Stream 1 18 94
STAND PROUD 4 38 90
End of the World 1 39 91
CRAZY NOISY BIZARRE TOWN 1 17 69
Chase 1 15 80
Great Days 1 23 88
Fighting Gold 1 31 96
Traitor's Requiem 1 15 89
Traitor's Requiem Third ver. 1 30 99
STONE OCEAN 1 19 79
Heavens Falling Down 2 01 60
Roundabout 8 35 44
It's Raining Somewhere Else 3 12 31
CORE 2 20 56
Heartache 1 48 31
THE WORLD REVOLVING 1 41 60
..... //Sigue hasta llegar a 100

```

Con un archivo de entrada como este esta mas que garantizado que siempre tengamos canciones disponibles para todo momento del dia.



The screenshot shows a window titled 'file' with a list of scheduled TV programs. The list includes various show titles, their start times, and broadcast channels (C for cable, P for pay-per-view). The shows listed are:

- 17:38:35 C Its TV Time!
- 17:41:26 C The Hero's Come Back!!
- 17:45:12 C Heavens Falling Down
- 17:48:14 C The Reluctant Heroes
- 17:52:00 C Zetsu Zetsu
- 17:54:15 P Movistar
- 17:54:36 P Nvidia GeForce
- 17:55:02 C My Dearest
- 17:58:58 P Nvidia GeForce
- 18:00:04 S El misterio de CROW Sesenta-y-Cuatro
- 18:16:32 P Movistar
- 18:16:53 C The Second Sanctuary
- 18:20:09 S El misterio de CROW Sesenta-y-Cuatro
- 18:36:37 P Nvidia GeForce
- 18:36:53 C A Cruel Angel's Thesis
- 18:40:59 S El misterio de CROW Sesenta-y-Cuatro
- 18:57:27 P Nintendo
- 18:57:58 C Kimi to no Yakusoku o Kazoeyo
- 19:01:44 P Nintendo
- 19:02:15 P Movistar
- 19:02:56 C STONE OCEAN
- 19:04:16 C Re:Re:
- 19:08:02 C Chase
- 19:09:18 C Blue Bird
- 19:13:04 C Blue Exorcist
- 19:16:50 P Nvidia GeForce
- 19:17:46 C Hologram
- 19:21:32 P Nintendo
- 19:22:03 C GUARDIAN
- 19:26:04 C Heartache
- 19:27:53 P Movistar
- 19:28:14 C Brave Heart
- 19:32:57 C Again
- 19:36:43 P Nvidia GeForce

Figura 16. Comportamiento del programa alrededor de la tercera franja estelar.

Puede observarse un flujo mucho más natural y sobre todo realista, mayormente en los bloques en los que se encuentran los escasos shows que fueron programados, a pesar de que sigue siendo algo repetitivo ver siempre las mismas 3 publicidades.

Es notable destacar que a pesar de tener mucho más contenido de mayor importancia a su disposición, el programa no deja de tener en cuenta a las publicidades, y se asegura de que abunden a lo largo del día, esto como medida para siempre intentar cumplir las reproducciones mínimas, sin importar que tan grande o pequeño sea el objetivo.

Por supuesto, fuera de los bloques en los que se encuentran los shows, se sigue observando comportamiento algo repetitivo, esto porque son sólo combinaciones distintas de canciones y cuñas, pero esto podemos arreglarlo incrementando la cantidad de shows a nuestra disposición:

-Varios shows, varias publicidades, varias canciones (Mejor Caso)

Con los siguientes archivos de entrada:

```
file  
//shows.in  
11  
Range Time with Franklin Cooper 10 15 5 7  
El misterio de CROW Sesenta-y-Cuatro 16 27 4 9  
The Escapists Audio Walkthrough Full Series 45 58 3 4  
MumboJumbo Redstone Tutorials 5 25 6 7  
A comprehensive Analysis of Robloxian Anatomy 15 23 4 7  
The Steel Ball Run Race 21 48 4 8  
Digimon World: Guia de Evolucion 19 25 4 6  
Cocina Facil con Minecraft 35 11 3 6  
PasaPalabra 5 47 8 6  
Geometry Dash guia del editor 59 58 2 7  
Tier list de tier lists 15 14 4 7
```

```
file  
//publicidad.in  
Nvidia GeForce 15 8  
Nintendo 30 4  
Movistar 20 10  
Cafe Madrid 22 12  
Ford Explorer 35 10  
Tecno SPARK 15 8  
Proton VPN 8 5
```

11 shows, 7 publicidades y las 100 canciones anteriores

The screenshot shows a software window with a light gray background. At the top left are three small circular icons. In the top right corner, the word "file" is written in a small, sans-serif font. The main area contains a list of scheduled events, each consisting of a timestamp, a letter indicating the source (S for song, P for program), and a title. The events are as follows:

- 06:51:09 C Haruka Kanata
- 06:54:55 P Nintendo
- 06:55:26 P Proton VPN
- 06:55:35 P Cafe Madrid
- 06:55:58 C Faint
- 06:58:47 P Tecno SPARK
- 06:59:03 P Ford Explorer
- 07:00:09 S Range Time with Franklin Cooper
- 07:10:25 P Nvidia GeForce
- 07:10:41 C Polygon Zone
- 07:13:39 S Range Time with Franklin Cooper
- 07:23:55 P Proton VPN
- 07:24:04 C STAND PROUD
- 07:28:43 S Range Time with Franklin Cooper
- 07:38:59 P Nintendo
- 07:39:30 C Bloody Stream
- 07:40:49 S Range Time with Franklin Cooper
- 07:51:05 P Movistar
- 07:51:26 C No.1
- 07:55:12 S Range Time with Franklin Cooper
- 08:05:27 P Nvidia GeForce
- 08:05:43 P Cafe Madrid
- 08:06:06 P Movistar
- 08:06:27 P Nvidia GeForce
- 08:06:43 P Cafe Madrid
- 08:07:06 P Nintendo
- 08:07:37 C Niji
- 08:11:55 S MumboJumbo Redstone Tutorials
- 08:17:21 P Tecno SPARK
- 08:17:37 S MumboJumbo Redstone Tutorials
- 08:23:03 P Proton VPN
- 08:23:12 S MumboJumbo Redstone Tutorials
- 08:28:38 C One Reason
- 08:32:24 S MumboJumbo Redstone Tutorials
- 08:37:50 P Ford Explorer
- 08:38:26 S MumboJumbo Redstone Tutorials
- 08:43:52 P Movistar
- 08:44:13 S MumboJumbo Redstone Tutorials
- 08:49:39 P Cafe Madrid
- 08:50:02 P Cafe Madrid

Figura 19. Flujo del programa alrededor del primer bloque estelar.

Este es el caso en el que mejor se desempeña el software, al tener a su disponibilidad una cantidad modesta de cada evento, es capaz de esparcirlos a lo largo del día para generar una programación realista. Son muy escasos los bloques extendidos de publicidad, porque ahora el programa tiene canciones suficientes para tapar ese tipo de huecos. Así como ahora dispone de bastantes shows distribuidos a lo largo del día para evitar que la emisora se vuelva una simple corneta de música.

En todos los casos de prueba, el programa logró cumplir las restricciones principales: separación de un segundo, no repetición de shows, cumplimiento del mínimo de publicidades y distribución adecuada según rating. Esto confirma la solidez de las funciones de control y la coherencia de los sorteos implementados.

Conclusiones

-Reflexión sobre la experiencia de desarrollo:

Lanzarse de lleno a desarrollar este proyecto sin tener ningún tipo de experiencia previa haciendo algo similar ha sido definitivamente una experiencia tanto traumática como cultivadora, fue muy intimidante el siqueira comenzar a desarrollarlo ya que no sentía que mis capacidades estuvieran a la altura, a pesar de tener una idea de hacia dónde se dirige el enunciado y el propósito del proyecto, desde un principio me hallaba frente al mismo problema, no es el que vas a hacer, sino cómo lo vas a hacer, como lo implementas, qué técnicas utilizas, donde haces X , donde haces Y, que haces en Z. Tuve que buscar ayudas externas, más de las que me hubiera gustado utilizar, poco a poco fui entendiendo sobre la marcha como debía implementar cada cosa. Aun así muchísimas veces me he encontrado completamente perdido respecto a cómo avanzar, cómo solucionar errores sin tener que reescribir todo el código y correr el riesgo de romper algo más, cada vez que hacia una nueva iteración del programa y resultaba ser un enfoque erróneo, me generaba un sentimiento de que estaba tirando todo el esfuerzo a la basura.

Eventualmente logré desarrollar el programa en su totalidad, y sabiendo como término, si pudiera volver y hacerlo de nuevo, **definitivamente** intentaría utilizar un enfoque diferente.

-Dificultades encontradas y soluciones:

Al principio del desarrollo cada vez que imprimía un horario todos los eventos (aunque me paso mucho más seguido con los shows) se apilaban al inicio del mismo, dejando desde mediados de la tarde hasta la noche simplemente vacíos y ya. Esto pasaba porque el programa insertaba un show cada vez que podía, cosa que me costó mucho lograr solucionar, tanto así que base el resto de la lógica de inserción, alrededor de los shows, intentando lo más posible que fueran esparcidos a lo largo del día es por eso que la función insertarShowDistribuido() es la más densa de todo el código.

También tuve varios errores almacenando los datos para consulta del usuario, principalmente porque al principio opte por leer los datos directo desde el archivo de salida, cosa que en su momento solo sobre complicaba excesivamente el manejo de dicho archivo, provocando que se sobreescribiera o cerrara antes de tiempo, así que al final, opte por almacenar todo en un arreglo aparte.

El programa en sí tuvo alrededor de 4 versiones, en todas y cada una de ellas tuve problemas organizando las funciones de inserción, condiciones no aplicables a todos los casos, algoritmos de funcionamiento deficiente de cara a la práctica, mal paso de variables por referencia entre funciones, ciclos infinitos, y en general una organización del código bastante deplorable, cada vez que empezaba a sobre complicar una función simplemente pasaba a un nuevo archivo y volvía a intentarlo de cero.

-Posibles mejoras:

Actualmente el programa genera grillas de manera completamente aleatoria, exceptuando a los shows estelares que siempre serán los 3 de mayor rating pero en orden variable. Si bien esto cumple con el enunciado previsto, a la hora de plantearse el realismo de este horario, si bien es algo completamente plausible, también es algo que puede mejorarse bastante.

El uso de 'bloques' de horario podría ayudar a que ciertos eventos se mantengan en horas estipuladas (Ej. un maratón de canciones de 3 a 6 PM), igualmente la implementación de algún tipo de control rating - horario tambien podria ayudar a aportar un poco más de realismo, actualmente la canción más popular de la emisora no tiene problema con ser reproducida a las 4 de la madrugada, hora en la que en la vida real muy poca gente está escuchando la radio y el show con el rating mas pesimo puede aparecer a las 3 o 4 de la tarde haciendo que todos los oyentes sientan la necesidad de cambiarse de canal, muy probablemente.

Un control más robusto y más dependiente de la hora del día, no tanto de la aleatoriedad ponderada por rating, podría ayudar a hacer mejores horarios de radio (también aumentaría en gran medida la complejidad del programa).

Si bien estoy orgulloso de lo que pude hacer con este programa, reconozco que tiene partes que podrían mejorarse, si tuviera que hacerlo, lo empezaría de cero con un enfoque completamente distinto, y trataría de tener una forma de codificar un poco más ordenada y menos espontánea. Aun así, ha sido una experiencia gratificante completar este desarrollo.