UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

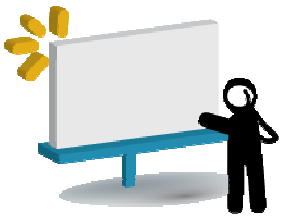
FUNDAMENTOS DE LA PROGRAMACIÓN

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES | Centro de e-Learning

MÓDULO 2 - UNIDAD 7

Integración de conceptos



Presentación:

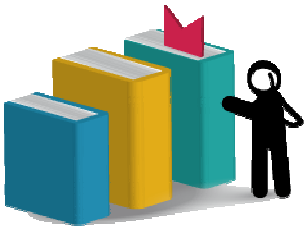
En esta Unidad nos tomaremos un descanso con respecto a incorporar temas y conceptos nuevos con el fin de analizar y vincular en un ejemplo práctico TODOS los temas vistos hasta el momento.

Objetivos:

Que los participantes:

- Comprendan cómo se relacionan y utilizan todos los temas vistos hasta el momento: variables, funciones, tipos de datos, arreglos, algoritmos, asignaciones, contadores, etc. ...

Bloques temáticos:



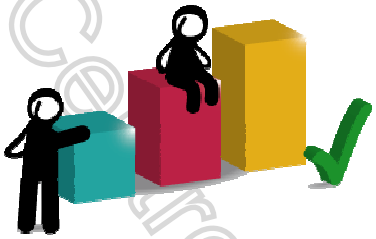
1. Programa integrador

2. Explicación del programa

2.1 Flujo del programa

2.2 Formas abreviadas

2.3 Asignaciones y otras instrucciones complejas



Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

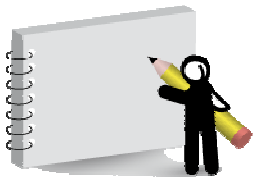
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

* El MEC es el modelo de E-learning colaborativo de nuestro Centro.



Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Programa integrador

En esta Unidad tomaremos un enfoque ligeramente distinto a la dinámica presentada en el curso, hasta ahora.

Antes de avanzar con conceptos más avanzados y complejos, resulta imprescindible integrar los temas vistos hasta el momento y aprovechar la ocasión para analizar algunas variantes a temas ya vistos y para poder ver, en un programa algo más extenso,

Los desafíos serán, entonces:

- Integrar y asentar los conceptos vistos, para estar preparados para seguir adquiriendo nuevos conocimientos

- Lograr poder seguir un programa extenso, sin perderse

Para lograr esto, analizaremos un programa dedicado administrar un VIDEOCLUB.

A los estudiantes más jóvenes, les recomiendo que lean esto antes de seguir:

https://es.wikipedia.org/wiki/Videoclub

:)

Este VIDEOCLUB hasta la fecha veía administrando las reservas con una planilla, a mano, pero su creciente clientela se está volviendo un factor determinante para replantear sus procesos de negocio. Una de las decisiones estratégicas de su propietario será contratar un desarrollado de software para que implemente un nuevo sistema que reemplace los procesos manuales, especialmente la gestión de las reservas de películas.

Un detalle importante es que el dueño del VIDEOCLUB (el cliente) tiene nociones de programación, aunque no tiene suficientes conocimientos (ni el tiempo) para programar su propio software.

Teniendo esto en cuenta, contra a un programador y le transmite sus requerimientos, expresando lo siguiente:

- "El alcance del software, para la Fase 1 del proyecto, deberá permitir: cargar películas en el catálogo, listar el catálogo de películas y alquilar y devolver películas"

- "Dado que conozco de programación, antes de contratar el desarrollo del software completo, quiero ver el programa desarrollado en pseudocódigo, para poder hacer ajustes y cambios (si es necesario) antes de tener el software funcional"

Por supuesto que aceptamos el desafío y comenzamos con el pseudocódigo. Luego de un tiempo, tenemos nuestra primera versión. Pactamos una reunión con nuestro cliente y le presentamos el programa.

Para facilitar la compresión (es un programa extremadamente extenso!), lo dividiremos en bloques:

- Encabezado: con el inicio del programa

- Función A, función B, ...: cada función separada

- Pie: con la finalización del programa

- **ENCABEZADO**

```
programa VideoClub
inicio
    //10 películas máximo, con sus datos: nombre, director, disponible      // ("si" o "no")
    //10 filas y 3 columnas
    var string peliculas[10][3]
    var integer cantidadPeliculas = 0
```

- **FUNCIÓN "PRINCIPAL"**

```
funcion principal()
    var integer opcion = 0
    mostrar: "Menu"
    mostrar: "----"
    mostrar: "1. Alta"
    mostrar: "2. Listar todas"
    mostrar: "3. Buscar película"
    mostrar: "4. Alquilar"
    mostrar: "5. Devolver"
    mostrar: "Ingrese opción"
    ingresar: opcion

    //verifico que se haya ingresado un número entre 1 y 5
    si opcion > 0 y opcion < 6 entonces
        en caso de opcion
            caso = 1
                si altaPelicula() = verdadero entonces
                    mostrar: "Pelicula creada OK"
                sino
                    mostrar: "No se pueden ingresar más películas"
                fin si
            fin caso
            caso = 2
                listarPeliculas()
            fin caso
            caso = 3
                buscarPelicula(ingresarPelicula())
            fin caso
            caso = 4
                alquilarPelicula(ingresarPelicula())
            fin caso
            caso = 5
                devolverPelicula(ingresarPelicula())
            fin caso
        fin en caso de
    fin si
fin funcion
```

- **FUNCIÓN "LISTAR PELICULAS"**

```
funcion listarPeliculas()
    var integer i = 1
    //recorro la matriz usando una variable auxiliar "i",
    //mientras i sea menor o igual a la
    //cantidad de películas que hay en la matriz
    mientras i <= cantidadPeliculas
        mostrar: "Nombre: " + peliculas[i][1] + " - Director: "
            + peliculas[i][2] + " - Disponible?: " + peliculas[i][3]
        i = i + 1
    fin mientras
fin funcion
```

- **FUNCIÓN "BUSCAR PELICULA"**

```
funcion integer buscarPelicula(String peliculaParaBuscar)
    var integer i = 1
    var boolean salgo = falso
    var integer posicion = 0

    mientras i <= cantidadPeliculas Y salgo = falso
        si peliculaParaBuscar = peliculas[i][1] entonces
                salgo = verdadero
                posicion = i
        fin si
        i = i + 1
    fin mientras
    retornar: posicion
fin funcion
```

- **FUNCIÓN "ALTA PELICULA"**

```
funcion boolean altaPelicula()
    //compruebo que la matriz no esté llena
    si cantidadPeliculas = 10 entonces
        retornar: falso
    fin si

    //incremento la cantidad de peliculas que hay en la matriz
    cantidadPeliculas = cantidadPeliculas + 1

    mostrar: "Ingrese nombre "
    ingresar: peliculas[cantidadPeliculas][1]
    mostrar: "Ingrese director "
    ingresar: peliculas[cantidadPeliculas][2]
    peliculas[cantidadPeliculas][3] = "si"

    retornar: verdadero
fin funcion
```

- **FUNCIÓN "INGRESA PELICULA"**

```
funcion string ingresarPelicula()
    var string nombrePelicula = ""

    mostrar: "Ingrese nombre pelicula "
    ingresa: nombrePelicula

    retornar: nombrePelicula
fin funcion
```

- **FUNCIÓN "ALQUILAR PELICULA"**

```
funcion alquilarPelicula(String nombrePeliculaAAlquilar)
    var integer peliculaAAlquilar = 0

    peliculaAAlquilar = buscarPelicula(nombrePeliculaAAlquilar)

    si peliculaAAlquilar = 0 entonces
        mostrar:"No se encontró la película buscada"
    sino
        si peliculas[peliculaAAlquilar][3] = "no" entonces
            mostrar:"La película ya se encuentra alquilada"
        sino
            peliculas[peliculaAAlquilar][3] = "no"
        fin si
    fin si

fin funcion
```

- **FUNCIÓN "DEVOLVER PELICULA"**

```
funcion devolverPelicula(String nombrePeliculaADevolver)
    var integer peliculaADevolver=0
    peliculaADevolver = buscarPelicula(nombrePeliculaADelvolver)

    si peliculaADevolver = 0 entonces
        mostrar:"No se encontró la película buscada"
    sino
        si peliculas[peliculaADevolver][3] = "si" entonces
            mostrar:"La película ya se encuentra disponible"
        sino
            peliculas[peliculaADevolver][3] = "si"
        fin si
    fin si
fin funcion
```

- **PIE**

```
fin
```

Fin.

Fin...?

Nuestro cliente terminó de leer el pseudocódigo y quedó boquiabierto, sólo llegó a decir:

- "Parece que no sé tanto de programación como pensaba que sabía..."

Con extrema paciencia, volvemos al principio del programa y explicaremos algunos puntos importantes, que seguramente clarifiquen el panorama de nuestro cliente.

2. Explicación del programa

2.1 Flujo del programa

El programa se inicia, como definimos, siempre en la "funcion principal". Pero antes de esto se declaran las variables globales del programa. Estas variables globales, al estar definidas AFUERA de la "funcion principal" puede ser accedidas y modificadas en cualquier punto del programa, en cualquier función.

Como estamos trabajando con funciones, sabemos que no puede haber NINGÚN código afuera de una función, salvo lo arriba mencionado de las variables globales.

Entonces bien, el programa se inicio en la "funcion prinicipal":

- Inicialmente se declaran algunas variables (locales, de esa función) y luego se muestra un menú de opción.

- El usuario ingresa valor y se VALIDA que esté dentro de la opciones permitidas del menú. Esto se hace en: "si opcion > 0 y opcion < 6 entonces".

    - Esta validación está presentada de una manera en particular, como vemos, pero también sería posible, y correcto, por ejemplo, hacerlo así:

        - "si opcion >= 1 y opcion =< 5 entonces"

        - O "si opcion =< 5 y opcion > 0 entonces" (aunque no sería muy elegante...)

- Comprobada la validación anterior, el programa va a tomar un único camino posible, dependiendo del número (opción del menú) ingresado. En este caso se va ejecutar alguna de las 5 alternativas presentadas en la estructura "en caso de".

- Una vez finalizada la opción elegida, el programa llega a su fin.

La clave para seguir el programa es:

- Imaginar qué opción del menú ingresa el usuario (será del 1 al 5 y la sugerencia será hacerlo en este orden ascendente)

- Ir leyendo de línea a línea

- Cuando se llega a una invocación o llamada a función, "saltar" a esa función y comenzar de nuevo, línea a línea en esa función. Una vez que se llega al "fin funcion" volver al mismo lugar desde donde se llamó a esa función

2.2 Formas abreviadas

Vemos algunas expresiones algo particulares. Estas expresiones realmente confundieron a nuestro cliente, por lo que será importante explicarlas paso a paso:

- "si altaPelicula()=verdadero entonces"

Sabemos que las funciones pueden o no devolver un valor. En caso en que una función devuelva un valor, literalmente, se la puede tomar como si fuera una variable. Una variable con un funcionamiento propio y complejo, pero una variable en cuanto a que puede "contener" un determinado valor.

Esta expresión es una forma común de ahorrase algunas líneas de código: no es más que eso. La forma "larga" sería:

```
...
    var boolean valor           //declaro una variable auxiliar
    valor = altaPelicula()      //guardo en la variable auxiliar el valor que
                                //devuelve la función "altaPelicula"
    si valor = verdadero entonces   //hago la condición en base a la variable auxiliar
...
```

- "buscarPelicula(ingresarPelicula())"

En este caso lo que vemos es un ejemplo de "anidando" de invocaciones a funciones. Este tipo de expresiones siempre deben analizarse de la misma forma que las expresiones matemáticas: desde adentro hacia afuera. Esto quiere decir que primero deberemos ver qué es lo que hace la función "ingresarPelicula" y luego analizar la función "buscarPelicula". Esta también es una forma de ahorrarse algunas líneas de código, por lo que podríamos ver cómo es la versión larga de la resolución:

```
...
    var string nombrePeli = ""          // creo una variable temporal
    nombrePeli = ingresarPelicula()     // le asigno el valor que devuelve la func.
    buscarPelicular(nombrePeli)     // envío como parámetro la variable temp
...
```

Por lo que vemos arriba, nos estamos evitando el paso de crear una variable temporal, asignarle un valor y luego usar esa variable como parámetro o argumento de la función de búsqueda.

2.3 Asignaciones y otras instrucciones complejas

- "ingresar: peliculas[cantidadPeliculas][1]"

En este caso vemos como se le asigna un valor en una posición del matriz: "cantidadPeliculas" será nuestro índice, un contador que nos va a decir cuál es la fila de la matriz que tiene lugar para que podamos ingresar una nueva película.

Recordemos las matrices tienen dos índices: matriz[i][j]. por convención, "i" va a ser el índice que indique las filas (posiciones horizontales) y "j" nos va a indicar las columnas (posiciones verticales). En este caso, cada película estará compuesta por 3 datos (nombre, director y disponibilidad). Esos 3 datos se van a guardar en cada fila. Por lo tanto, si quiero agregar una película, voy a llenar una fila de la matriz con estos 3 datos. Por eso, "cantidadPeliculas" no cambia. Lo que SÍ cambian son las columnas donde van los 3 datos. En la columna 1 va el nombre de la película. El columna 2 el director y en la 3 la disponibilidad. Por eso es que los datos de las películas se van a guardar en:

- peliculas[cantidadPeliculas][1]: Acá va el nombre

- peliculas[cantidadPeliculas][2]: Acá va el director

- peliculas[cantidadPeliculas][3]: Acá va la disponibilidad

Lo mismo ocurre cuando quiero leer o mostrar un dato. Para eso, voy a tener que recorrer la matriz con un nuevo índice (un contador) e ir mostrando columna a columna. Es lo que se ve en "peliculas[i][1]", donde "i" es el contador y el número "1" representa la primer columna (o sea, en este caso, vamos a consultar el nombre de la película en la fila "i").

- "mientras i <= cantidadPeliculas Y salgo=falso"

Este es un ciclo repetitivo con doble condición. Esto significa que el contenido de esta estructura se va a ejecutar tantas veces como ambas condiciones se cumplan. AMBAS CONDICIONES EN SIMULTÁNEO.

En este caso, la condición de salida del ciclo se cumplirá cuando:

- "i <= cantidadPeliculas": el contador "i", que se incrementa dentro de este ciclo, sea menor o igual que el índice "cantidadPeliculas". Recordemos que "cantidadPeliculas" nos va a decir siempre, en todo momento y lugar del programa, cuántas películas tenemos en nuestro catálogo. Por lo cual, siendo "i=1" y "cantidadPeliculas = 5", el ciclo se va repetiría 5 veces.

Pero como tenemos una doble condición, vamos a tener que analizar qué sucede con la segunda para terminar de entender la condición de salida del ciclo mientras.

- "salgo=falso": la "bandera" (flag) o marcador "salgo" se define como "salgo=falso" al inicio de la función. Esto significa que a la primera iteración del ciclo mientras, esta condición será verdadera. Es un poco confuso, pero vale la pena pensarlo un segundo: "si salgo=falgo, entonces es verdad". No es que "salgo" sea verdadera, sino la instrucción "salgo=falso" será verdadera, ya que ***la condición evalúa a toda la instrucción y no la variable en sí***. El valor de "salgo" es modificado dentro del ciclo. En caso en que su valor sea modificado, esta segunda condición dejará de ser verdadera y el ciclo se interrumpirá.

Teniendo ambas condiciones en claro, el ciclo se ejecutará tantas veces como AMBAS sean verdaderas. En otra palabras, la explicación en modo narrativo, sería: **"el ciclo mientras se va a ejecutar tantas veces como no se haya recorrido el catálogo completo y que variable "salgo" mantenga su valor inicial"**.

Y, ¿porqué uso una condición doble en el mientras?

En nuestro caso, sólo vamos a poder tener un catálogo de 10 películas, porque la matriz tiene sólo 10 filas. Pero supongamos el catálogo de algún servicio de streaming pago de películas, que probablemente tenga miles de registros. La primera condición (lo veremos con más profundidad en la próxima Unidad) realiza lo que se conoce como "búsqueda secuencial". Esto significa que se sitúa al comienzo de la matriz (o vector) y va pasando fila por fila, celda por celda, comparado el valor buscado con el valor de cada fila o celda. Ahora, supongamos que tenemos 100.000 películas, y que la película buscada está en la fila número 5. En ese caso, la búsqueda encuentra en un resultado positivo a la quinta iteración... pero el ciclo continuaría, y lo repetiría 99.995 veces más... sin ningún sentido, porque la búsqueda ya fue exitosa en el ciclo número 5. Esto es un claro ejemplo de una mala programación, porque no se tienen en cuenta los recursos que se utilizan. ¿Hay algún error? No, seguro que no. Hasta sería un programa que compila y funciona. Pero sería muy ineficiente y hasta negligente hacer un programa de este tipo, sin la menor consideración por los recursos utilizados.

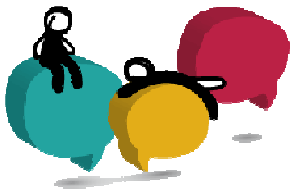
Entonces, y ya que tenemos que recorrer secuencialmente la matriz, lo que podemos hacer es agregar un doble condición, que serviría como "gatillo" en caso de encontrar un resultado y poder interrumpir el ciclo en forma prematura y controlada.

Como podrán ver, cuando la película buscada es encontrada (), se le cambia el valor al segundo término de la condición ("salgo=verdadero"), por lo que a la próxima iteración, cuando se evalúa de nuevo esa condición", está será FALSA, por lo tanto, ambas condición en su conjunto dejarán de ser verdaderas. Esto se debe a que ambos términos de la condición doble está unidos a través del operador lógico ""Y".

De esta forma, si tenemos 100.000 películas, y la película buscada está en la posición 5, 400 o 87.231, el ciclo mientras se ejecutará 5 veces, 400 veces u 87.231 veces, respectivamente. La única forma en que el ciclo se repita 100.000 veces, será en el caso en que la película buscada esté en la última posición, o sea, en la posición cien mil.

- "peliculas[peliculaADevolver][3] = "si""

Recordemos que en nuestra tercer columna de la matriz vamos estar guardando la disponibilidad de cada película para saber si está alquilada o si puede ser alquilada. Para esto, en dicha columna guardaremos un "si" si la película se encuentra disponible de ser alquilada o un "no" si la película no está disponible ya que fue alquilada previamente. En este caso "peliculaADevolver" es una variable tipo índice que guarda la posición (fila) de una película a que la estamos marcando como disponible, al asignarle un "si" es su tercer columna.



ACTIVIDAD DE ANÁLISIS:

Luego de analizar el código del programa del VIDEOCLUB en detalle, analizar los siguientes requerimientos:

Requerimiento 1. El principal problema que tiene ese video club es que su software de gestión no puede hacer más de una cosa a la vez. O sea: se listan las películas, el programa termina. Se agrega una película, el programa termina. Se alquila una película... El requerimiento del cliente es hacer los cambios en el programa necesarios para poder ejecutar todas las funciones que el usuario desee y que el programa termine cuando el usuario lo decida.

Requerimiento 2. El cliente quisiera saber quién alquiló cada película. Para esto debería registrar el nombre del socio que se lleva cada película y tener una forma de buscar, por nombre de socio, qué películas tiene alquiladas en simultáneo un socio en particular.

Requerimiento 3. El dueño del VIDEOCLUB necesita conocer la recaudación de la semana. Para esto se requiere una nueva función que permita ir cargando los ingresos totales por día de la semana y, una vez finalizada la carga, que muestre un mensaje con la recaudación total.

Recomendaciones:

- Modificar lo mínimo imprescindible para cumplir con los requerimientos. No agregar cambios que no aporten una solución o no sean parte de la solución concreta.

- Sobre el Requerimiento 1: agregar solamente una opción nueva al menú no es suficiente para que se cumpla con el requerimiento

- Sobre el Requerimiento 2: Se recomienda enfáticamente NO agregar arreglos (matrices o vectores) salvo que sea absolutamente necesario

- Sobre el Requerimiento 3: tengan en cuenta de NO mostrar valores parciales, solo pide el total semanal.

Podrán comparar sus respuestas con los ejemplos resueltos en el "ANEXO 1 - Respuesta Actividad de Análisis" al final de esta unidad.

Previamente pueden consultar el "ANEXO 2 - Checklist de código" de la unidad anterior.

IMPORTANTE:

- **¡Traten de resolverlo cada uno por sus propios medios!**
- **Las respuestas (el programa) NO debe ser enviado a los foros del Campus, salvo dudas puntuales de las respuestas.**

ANEXO 1 - Respuesta Actividad de Análisis

¿Trataron de resolver los requerimientos?

En caso afirmativo, avancen 3 páginas.

En caso negativo, ¡no avancen y traten de resolverlos antes de ver las respuestas!

Página dejada intencionalmente en blanco

Página dejada intencionalmente en blanco

Página dejada intencionalmente en blanco

```
programaVideoClub
inicio
    //agrego una columna para guardar el nombre, según requerimiento 2
    string peliculas[10][4]
    var integer cantidadPeliculas=0

    funcion principal()
        var integer opcion=0
        mostrar: "Menu"
        mostrar: "----"
        mostrar: "1. Alta"
        mostrar: "2. Listar todas"
        mostrar: "3. Buscar pelicula"
        mostrar: "4. Alquilar"
        mostrar: "5. Devolver"
        //nueva opción de menú, según requerimiento 2
        mostrar: "6. Alquileres por socio"
        //nueva opción de menú, según requerimiento 3
        mostrar: "7. Recaudación "
        //nueva opción de menú, según requerimiento 1
        mostrar: "999. Salir "
        mostrar: "Ingrese opcion"
        ingresar: opcion

        //manejo nuevos valores posibles "opcion", según requerimiento 1
        si opcion>0 y opcion<8 entonces
            en caso de opcion
                caso = 1
                    si altaPelicula()=verdadero entonces
                            mostrar: "Pelicula creada OK"
                    sino
                            mostrar: "No se pueden ingresar mas peliculas"
                    fin si
                fin caso
                caso = 2
                    listarPeliculas()
                fin caso
                caso = 3
                    buscarPelicula(ingresarPelicula())
                fin caso
                caso = 4
                    alquilarPelicula(ingresarPelicula())
                fin caso
                caso = 5
                    devolverPelicula(ingresarPelicula())
                fin caso
                //agrego caso para manejar opcion=6
                caso = 6
                    buscarAlquileresPorSocio()
                fin caso
```

```
            //agrego caso para manejar opcion=7
            caso = 7
                calcularYMostrarRecaudacion()
            fin caso
        fin en caso de
        //llamada recursiva, según requerimiento 1
        principal()
    //agrego validación de salida, según requerimiento 1
    sino si opcion = 999 entonces
        mostrar: "Cerrando..."
    //agrego validación de valor ingresador incorrecto, según requerimiento 1
    sino
        mostrar: "Menú incorrecto. Reintentar..."
        principal()
    fin si
fin funcion


funcion listarPeliculas()
    var integer i=1
    mientras i <= cantidadPeliculas
        mostrar: "Nombre: " + peliculas[i][1] + " -
          Director: " + peliculas[i][2] + " ? Disponible?: " + peliculas[i][3]
       i = i + 1
    fin mientras
fin funcion


funcion integer buscarPelicula(String peliculaParaBuscar)
    var integer i=1
    var boolean salgo=falso
    var integer posicion=0
    mientras i <= cantidadPeliculas Y salgo=falso
    si peliculaParaBuscar = peliculas[i][1] entonces
        salgo=verdadero
        posicion=i
    fin si
    i = i + 1
    fin mientras
    retornar: posicion
fin funcion


funcion boolean altaPelicula()
       si cantidadPeliculas = 10 entonces
              retornar: falso
       fin si
       cantidadPeliculas = cantidadPeliculas + 1
       mostrar: "Ingrese nombre "
       ingresar: peliculas[cantidadPeliculas][1]
```

```
        mostrar: "Ingrese director "
        ingresar: peliculas[cantidadPeliculas][2]
        peliculas[cantidadPeliculas][3] = "si"
        //inicializo, según requerimiento 2
        peliculas[cantidadPeliculas][4] = ""
        retornar: verdadero
fin funcion


funcion string ingresarPelicula()
    var string nombrePelicula=""
    mostrar: "Ingrese nombre pelicula "
    ingresa: nombrePelicula
    retornar: nombrePelicula
fin funcion


funcion alquilarPelicula(String nombrePeliculaAAlquilar)
    var integer peliculaAAlquilar=0
    peliculaAAlquilar = buscarPelicula(nombrePeliculaAAlquilar)
    si peliculaAAlquilar = 0 entonces
        mostrar:"No se encontr? la pel?cula buscada"
    sino
        si peliculas[peliculaAAlquilar][3] = "no" entonces
            mostrar:"La pel?cula ya se encuentra alquilada"
        sino
            peliculas[peliculaAAlquilar][3]="no"
            //guardo el nombre del socio que alquila, según requerimiento 2
            peliculas[peliculaAAlquilar][4] = ingresarSocio()
        fin sin
    fin si
fin  funcion


funcion devolverPelicula(String nombrePeliculaADevolver)
    var integer peliculaADevolver=0
    peliculaADevolver = buscarPelicula(nombrePeliculaADevolver)

    si peliculaADevolver = 0 entonces
        mostrar:"No se encontr? la pel?cula buscada"
    sino
        si peliculas[peliculaADevolver][3] = "si" entonces
            mostrar:"La pel?cula ya se encuentra disponible"
        sino
            peliculas[peliculaADevolver][3]="si"
            //elimino el nombre del socio que alquiló, según requerimiento 2
            peliculas[peliculaAAlquilar][4] = ""
        fin sin
    fin si
fin funcion
```

```
    //agrego función para reutizar esta porción de código
    funcion string ingresarSocio()
        var string socio = ""
        mostrar: "Ingresar nombre de socio"
        ingresar: socio
        retornar: socio
    fin funcion

    //agrego función de búsqueda, según requerimiento 2
    funcion buscarAlquileresPorSocio()
        var string socioABuscar
        var integer i = 1
        var integer cantidadAlquiladas = 0
        socioABuscar = ingresarSocio()

        mientras i <= cantidadPeliculas hacer
            si peliculas[i][4] = socioABuscar entonces
                mostrar: "Alquiló: " + peliculas[i][1]
                cantidadAlquiladas = cantidadAlquiladas + 1
            fin si
            i = i + 1
        fin mientras

        si cantidadAlquiladas = 0 entonces
            mostrar: "El socio buscado no tiene ninguna alquilada"
        sino
            mostrar: "El socio buscado alquiló " + cantidadAlquiladas
        fin si
    fin funcion


    //agrego función de recaudacion, según requerimiento 3
    funcion calcularYMostrarRecaudacion()
        var integer i = 1
        var float acumulado = 0
        var float recauda[7]

        mostrar: "Dia: 1:lunes, 2:martes, 3:miercoles, 4:jueves, 5:viernes, 6:sabado, 7
             :domingo"

        mientras i < 8 hacer
            mostrar: "Ingrese la recaudación del día " + i
            ingresar: recauda[i]
            acumulado = acumulado + recauda[i]
            i = i + 1
        fin mientras

        mostrar: "el total de la semana es: " + acumulado
    fin funcion
fin
```

ANEXO 2 - Preguntas frecuentes

1) En la función buscarPelicula(ingresarPelicula()) se retorna la variable "posicion" con el número de fila en el que se encuentra la película buscada (i), pero en ningún momento se agrega código que permita visualizar cuál es ese número, como por ejemplo: "mostrar: posicion", o "mostrar: buscarPelicula(ingresarPelicula())". ¿Por qué?

Respuesta: En este caso no se está haciendo nada con el dato que retorna la función, lo cual no es un error de código sino, más bien, un error conceptual.

¿Qué podríamos hacer ahí? Se podría mostrar el número de la fila en menú, ver si está disponible, mostrar la información completa de la película, o ver simplemente indicar si existe o no una película.

Siempre, con un dato retornando la idea es usarlo. Pero si se usa, tampoco implica que haya error de código.

2) P: Cuando se inicializa una variable ¿el primer valor que se le asigna, es el valor más bajo que ésta puede llegar a tomar?

Por ejemplo: var integer cantidadPeliculas=0

Se le asigna 0 (cero), pero ¿por qué no se inicia en 1?

R: N, no se le da el valor "más bajo", ya que, por ejemplo, -1 es menor a 0. Esa no es la razón.

Las variables se inicializan con el primer valor que se va a usar. Por ejemplo, si es un contador que vas a usar para recorrer un vector, y los vectores inician en la posición 1, se inicializa la variable en 1.

En el ejemplo citado, al principio del programa hay "cero películas", por lo tanto es correcto inicializar el contador en cero.

3) P: Sobre la terminología correcta: está correcto tomar el término "inicializar" como sinónimo de "asignarle valor"?

R: Toda inicialización de variable implica hacer un asignación. Dicho de otra manera, la primera asignación va a ser la inicialización de la variable.

4) P: En la función "buscarPelicula", ¿por qué se retorna la variable "posicion" y no "i" que es la guarda la posición de las filas?

R: En "posicion" se guardar el "i" correcto. Por ejemplo:

- **Se encuentra la película buscada en la fila 6.**
- **Por lo tanto, "i" es 6.**
- **Se entra en la condición, se cambia "salgo" y se guarda en "posicion" el 6 que vale "i"**
- **LUEGO se incrementa "i" y pasa a valer 7**
- **Cuando se repite el "mientras" no se vuelve a entrar ya que la variable "salgo" cambió de valor.**
- **Luego "fin mientras".**
- **Luego retorna "posicion".**

Por lo tanto, al momento de finalizar la función:

- **"posición" vale 6, que es la fila dónde se encontró la película buscada**
- **"i" vale 7.**

5) P: Cuando se declara una matriz se especifica un único tipo de dato para toda la matriz. Como ocurre en el programaVideoClub, que se declara la matriz:

var string peliculas[10][3]

Por lo tanto, ¿todos los valores que se carguen en esa matriz, deben ser datos del tipo string? ¿Hay alguna posibilidad de tener distintos tipos de datos en una misma matriz?

R: Algunos lenguajes permiten la conversión de tipos de datos: se llama "casting" y es un tema que se presentará en este curso más adelante, aplicado a Programación Orientada a Objetos. De momento, existe el límite de usar un mismo tipo de dato para toda la estructura.

6) P: Sobre el Requerimiento 1: es posible invocar la función principal al final de cada función, para que el cliente vuelva a acceder al menú principal luego de haber realizado alguna de las opciones y no tenga que volver a abrir el programa como lo dice ahí.

R: No es que sea imposible, pero con llamar a la función "principal" a cada función no alcanza para cumplir con el requerimiento. Además, en el caso de la función "buscar..." sería un error, porque se rompen las funciones "alquilar..." y "devolver..." que usan esa función.

7) P: Sobre el Requerimiento 2: se recomienda enfáticamente no agregar arreglos, por lo tanto, ¿se puede modificar (agregar una columna) al arreglo ya existente?

R: Se pueden hacer todos los cambios que sean crean necesarios. Pero hay que tener en cuenta que la idea es tratar de hacer la MENOR cantidad posible de cambios para cumplir con los requerimientos.

8) P: Sobre el Requerimiento 3: ¿el "cliente" va a realizar el balance semanal un sólo día? ¿Al terminar la semana quiere colocar todos lo recaudado día por día y que el programa le dé el total de lo recaudado?

R: No está especificado, pero se podría considerar como un procesamiento válido.

9) P: ¿Cómo se debe interpretar la expresión salgo=falso? ¿Sería como un flag para indicar cuándo hay que salir de un bucle?

R: Las claves de la instrucción "salgo=falso" son:

- **Diferenciar el valor de la variable del resultado de la condición**
- **Notar con qué valor se inicializa la variable "salgo"**
- **Notar dónde cambia de valor la variable "salgo"**
- **Interpretar que el cambio del valor de la variable implica también que cambie el resultado de la condición**

Por otro lado, si, es una variable tipo "bandera" o "flag".

10) P: ¿Cómo se evitar tener una instrucción larga como esta?:

mostrar: i + " - " + pelicula[i][1] + " - Dir: " + pelicula[i][2] + " - " + "Disponibilidad: " + pelicula[i][3]

R: La sugerencia es hacer un corte de línea en cualquiera de los operadores.

Por ejemplo:

```
mostrar: i + " - " + pelicula[i][1] + " - Dir: " +
        pelicula[i][2] + " - " + "Disponibilidad: " + pelicula[i][3]
```

O

```
mostrar: i + " - " + pelicula[i][1] +
        " - Dir: " + pelicula[i][2] +
        " - " + "Disponibilidad: " +
        pelicula[i][3]
```

El segundo caso es una exageración, desde ya.

11) P: ¿Cuántas matrices puede haber dentro de un mismo programa?

R: Todas las matrices que se requieran, no hay límite preestablecido a nivel de lenguaje.

Lo que vimos

- Uso integrado de todos los temas vistos hasta el momento: variables, funciones, tipos de datos, arreglos, algoritmos, asignaciones, contadores...

Lo que viene:

- Estructuras de datos principales: listas (lineales y simples), pilas y colas
- Métodos de ordenamiento por selección, por inserción, por intercambio y por intercalación
- Métodos de búsqueda secuenciales y búsqueda binaria