UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

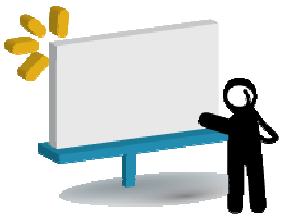
FUNDAMENTOS DE LA PROGRAMACIÓN

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:

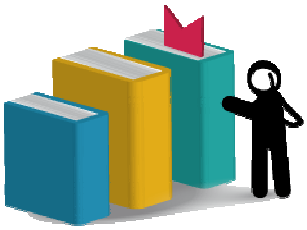
En esta Unidad nos tomaremos un descanso con respecto a incorporar temas y conceptos nuevos con el fin de analizar y vincular en un ejemplo práctico TODOS los temas vistos hasta el momento.

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:



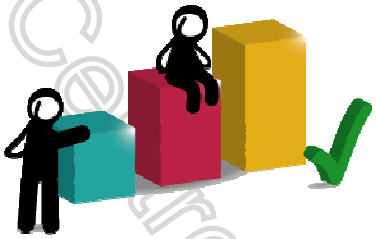
1. Programa integrador

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:

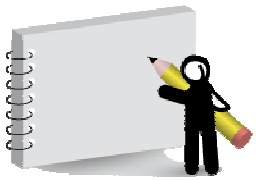
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

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.

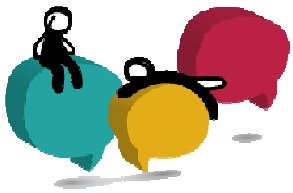
Entonces, y ya que tenemos que recorrer secuencialmente la matriz, lo que podemos hacer es agregar un doble condición, que serviría como "gatillo" en caso de encontrar un resultado y poder interrumpir el ciclo en forma prematura y controlada.

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.



Actividades de la Unidad 7

Luego de analizar el código del programa del VIDEOCLUB en detalle, en el foro de la Unidad, realizar los siguientes cambios a dicho programa:

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 generales:

- 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.

- Siempre es mejor entregar 1 requerimiento terminado que 3 requerimientos con errores o incompletos. Puede elegir cuál entregar, no es necesario que sean los 3 requerimientos.

Recomendaciones específicas:

- 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.

Condiciones de entrega:

- Escribir la solución en pseudocódigo y deben subir el código editable (no imagen).

- Pueden resaltar las partes nuevas o las partes que se modifican

- Cada uno deberá hacer la entrega en un único thread/tema del foro para poder mantener y tener a mano el historial de correcciones y reentregas.

Para hacerlo en un poco más realista, vamos a agregar la siguiente condición:

IMPORTANTE: esta actividad admite ÚNICAMENTE UNA (1) re-entrega (*)

(*) Entrega --> corrección --> reentrega --> corrección

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