


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

DIVISIÓN SISTEMA UNIVERSIDAD ABIERTA Y EDUCACIÓN A DISTANCIA

PROGRAMACION DE DISPOSITIV.MOVILES

UNIDAD 5

ACTIVIDAD TEORICA

NOMBRE DEL ALUMNO:

PAULINA RODRIGUEZ SAMPEDRO

NOMBRE DEL ASESOR:

CRISTIAN CARDOSO ARELLANO

SEXTO SEMESTRE

GRUPO: 9696

CONTROL DE FLUJO, FUNCIONES Y LAMBDAS

Objetivo

Conocer los conceptos y herramientas fundamentales de la lógica de programación para construir aplicaciones estructuradas, eficientes y fáciles de mantener.

Declaraciones condicionales

Estructura de control que permite al programa tomar decisiones y ejecutar diferentes bloques de código según una condición específica, actuando como un semáforo para dirigir el flujo del programa.

La sentencia if

Es la estructura más básica, y se usa para ejecutar un bloque de código solo si se cumple una condición.

Ejecución de código condicional con la instrucción if...else

Permite manejar dos caminos: uno si la condición es verdadera (se ejecuta el bloque if) y (else) si es falsa.

La declaración switch

Evalúa una expresión y ejecuta el bloque que coincide con el caso. Es decir, evalúa una expresión (por ejemplo, un número, un texto, o un valor) y luego ejecutar el bloque de código que corresponde al caso que coincide con ese valor.

El bucle for-in

Se usa para recorrer colecciones o rangos. Se usa cuando se sabe exactamente qué se desea recorrer, por ejemplo una lista de nombres, un rango de números o un diccionario.

El ciclo while

Ejecuta un bloque mientras la condición sea verdadera. Es decir, la condición se evalúa antes de entrar al bloque, si la condición es falsa desde el inicio, el código nunca se ejecuta.

Filtrado con la sentencia where

Es una extensión de control que se añade a un bucle (como el for-in) o a una estructura de selección (como el switch) para aplicar una condición lógica antes de ejecutar el código. Es decir, en lugar de procesar todos los elementos de una lista, el programa evalúa cada uno y solo permite la entrada al ciclo a aquellos que devuelvan un resultado verdadero en la comparación.

Filtrado con la instrucción for-case

Se usa para buscar un patrón específico dentro de una lista completa. Es decir, for recorre todo lo que hay en una lista. Pero con for-case, es como decirle al programa que solo entre al ciclo si el elemento tiene cierta forma específica.

Usando la sentencia if-case

Se usa cuando solo te interesa un caso específico de un valor, en lugar de usar un switch completo con todos los casos.Es decir, if-case se enfoca exclusivamente en el valor que interesa en ese momento. Si el valor coincide con el patrón, el código se ejecuta; si no, el programa simplemente continúa su curso sin necesidad de escribir casos adicionales o cláusulas default.

Declaraciones de transferencia de control

Son instrucciones que alteran el flujo normal de ejecución de un programa.

La declaración de break

La break instrucción tiene dos formas: etiquetada y no etiquetada. La forma no etiquetada se usa para salir inmediatamente de una estructura de control switch, y la forma no etiquetada para terminar un bucle for, whileo do-while.

Declaraciones condicionales

Son estructuras de control que permiten que una aplicación tome decisiones dependiendo de si una condición es verdadera o falsa.

Uso de una función de parámetro único

Es un método que recibe un solo dato para trabajar.

Uso de una función multiparámetro

Un método que recibe dos o más datos, separados por comas. El orden en que los envías debe coincidir con el orden en que se declararon.

Definición de los valores predeterminados de un parámetro

Son valores que se asignan a uno o más parámetros en la definición de una función. Es decir, si al momento de llamar a la función no proporcionamos un argumento para ese parámetro, la función no dará error; en su lugar, utilizará automáticamente el valor que definiste por defecto.

Agregar nombres de parámetros externos

Los parámetros externos son etiquetas que se crean para que, al momento de usar (llamar) la función, el código se lea como una oración natural, pero dentro de la función se usa un nombre distinto y más corto. Es decir, en la declaración de la función se colocará el nombre externo seguido del nombre interno. El nombre

externo sirve para dar claridad a quien usa la función, y el nombre interno sirve para que uno mismo trabaje con ese valor dentro de los corchetes.

Uso de pará metros variádicos

Permite que una función reciba una cantidad indefinida de datos del mismo tipo. Es decir, en lugar de que el usuario pase exactamente uno o dos valores, la función puede aceptar 0, 5, 10 o 100 valores sin que se tengas que cambiar el código.

Parámetros de entrada y salida

Los parámetros In-Out permiten modificar una variable externa directamente. Es decir, cuando se pasa una variable a una función, la función recibe una copia. Si se desea cambiar el valor dentro de la función, el valor original que está "afuera" no se entera y se queda igual.

Una introducción a las lambdas

Es una forma de escribir una función anónima (una función sin nombre) de manera extremadamente corta. Es decir, permiten tratar la funcionalidad como argumento de un método o el código como datos.

Sintaxis abreviada para lambdas

(parámetros) -> expresión_o_bloque_de_código

1. Los Parámetros (v)

Es la variable que recibe el método. En un botón de Android, el método onClick(View v) recibe una vista. En la Lambda, solo se coloca v.

2. La Flecha ->

Es el símbolo que le dice al compilador: "Lo que está a la izquierda son los datos, lo que está a la derecha es lo que debes hacer con ellos".

3. El Cuerpo { ... }

Es el código que antes se ponía dentro del public void onClick. Si es una sola línea, ni siquiera se necesitan las llaves {}.

Conclusión

El conocimiento de estos conceptos y estructuras es fundamental para el profesional de informática porque fortalece el pensamiento lógico y la base para gestionar la complejidad algorítmica. Además, proporciona la capacidad de diseñar soluciones técnicas que no solo funcionan, sino que son eficientes y escalables. Asimismo, este dominio técnico permite construir un código modular y mantenible mediante el uso correcto de funciones, parámetros y lambdas. Esto es crítico en entornos profesionales donde el trabajo en equipo y la evolución del software a largo plazo son la norma; un código bien estructurado reduce significativamente la "deuda técnica" y facilita la detección de errores.

Referencias

Oracle. (n.d.). *Lambda expressions*. In *The Java™ Tutorials: Learning the Java Language – Classes and Objects*. Oracle. https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html (docs.oracle.com

Oracle. (n.d.). *Branching statements*. In *The Java™ Tutorials — Learning the Java Language: Language Basics*. Oracle. https://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html

Oracle. (n.d.). *Passing information to a method or a constructor*. In *The Java™ Tutorials: Learning the Java Language – Classes and Objects*. Oracle. https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html