Universidad Tecnológica de Durango

Tecnologías de la Información

Fundamentos de programación

Actividades

“Evidencias de Activiades y Tareas”

Alumnos:

- Barraza Torres Jesús Daniel

1°A BIS

Docente:

- Ing. Dagoberto Fiscal Gurrola, M.T.I.

Febrero 2025

Tabla de Ilustraciones

Actividad 1

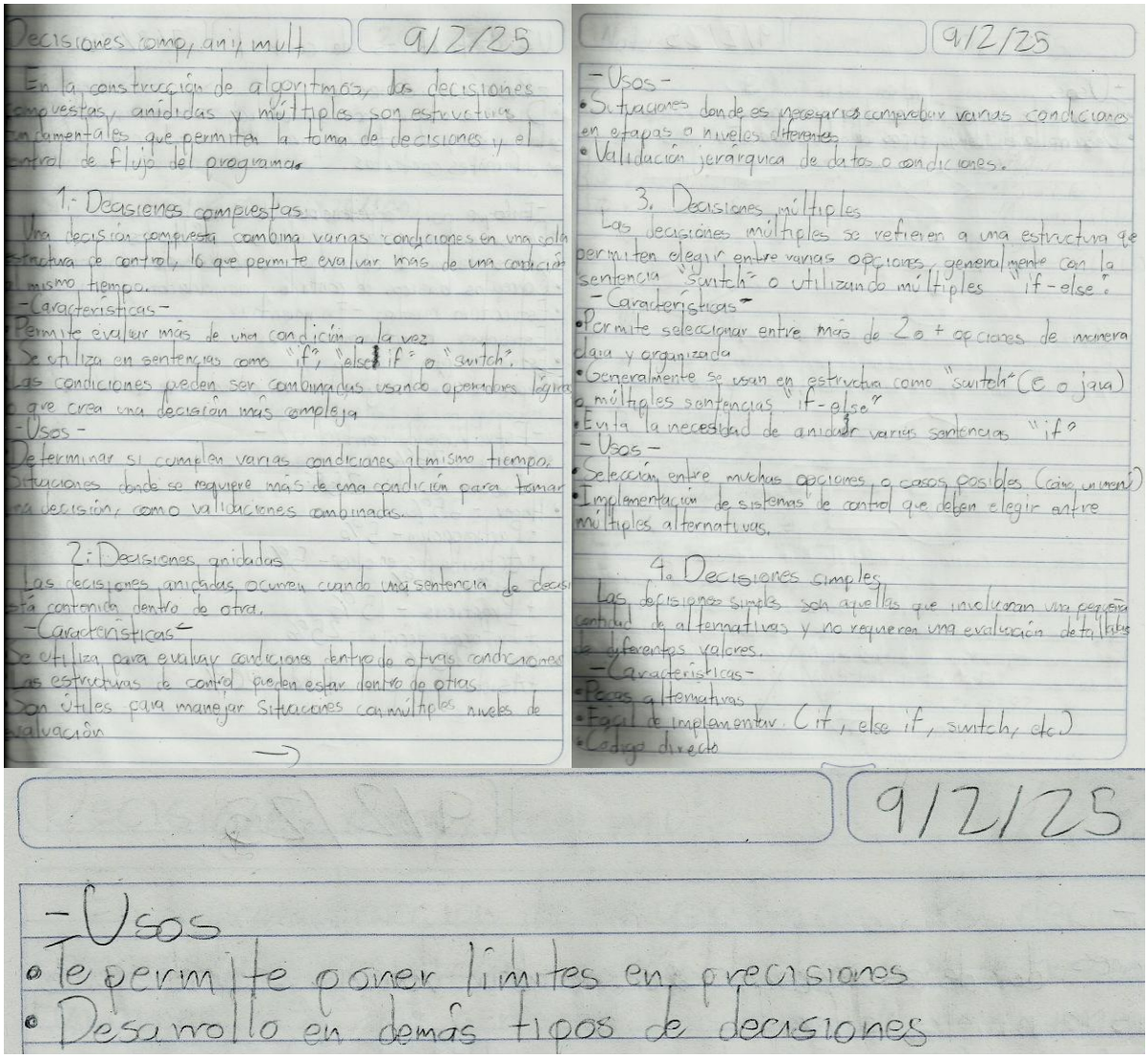


Ilustración 1. Decisiones compuestas, anidadas y múltiples

En la imagen anterior se muestran las 4 categorías de las decisiones en la programación: Simples, compuestas, anidadas y múltiples; así como sus características y sus usos posibles dentro de la programación.

Actividad 2

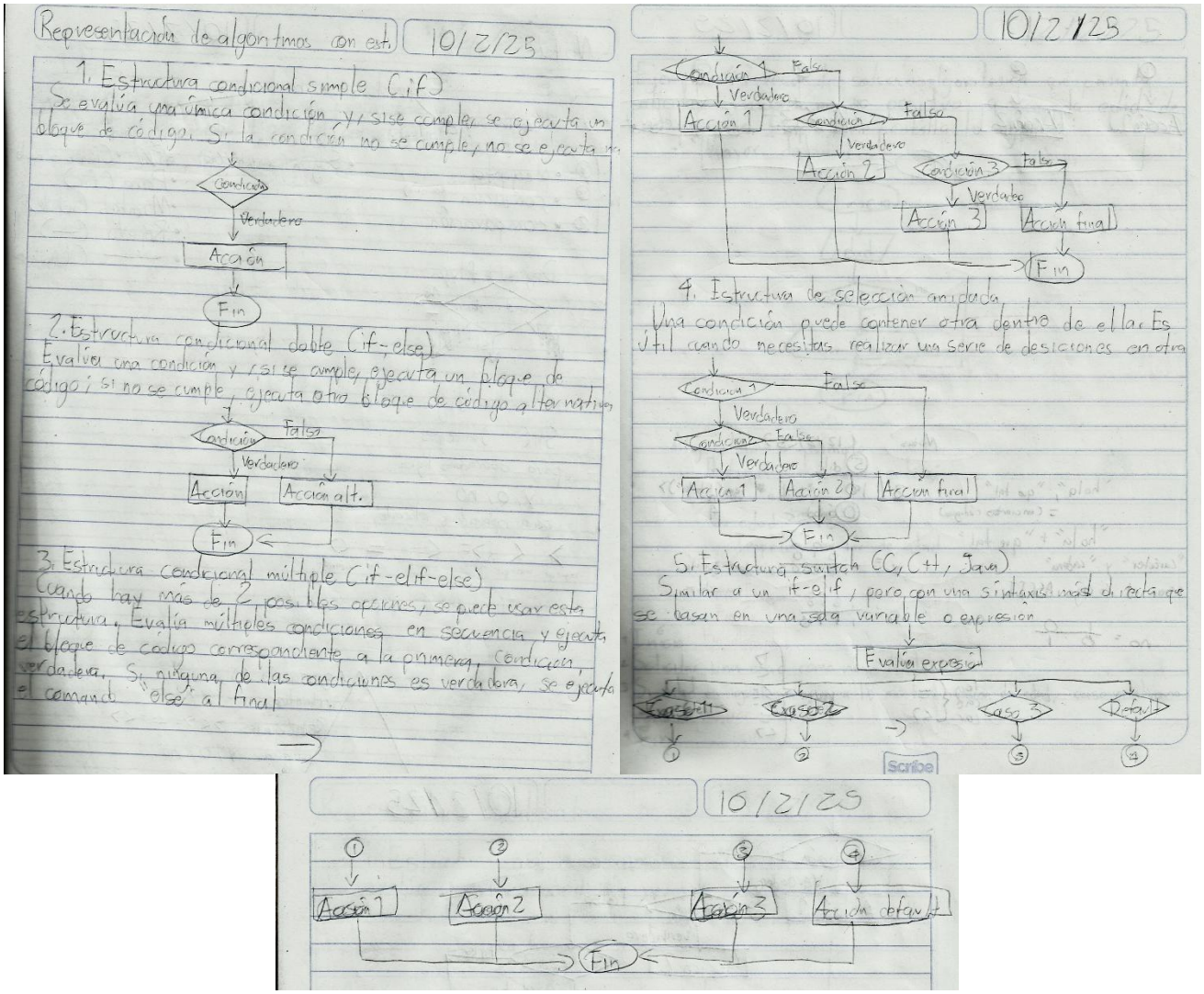
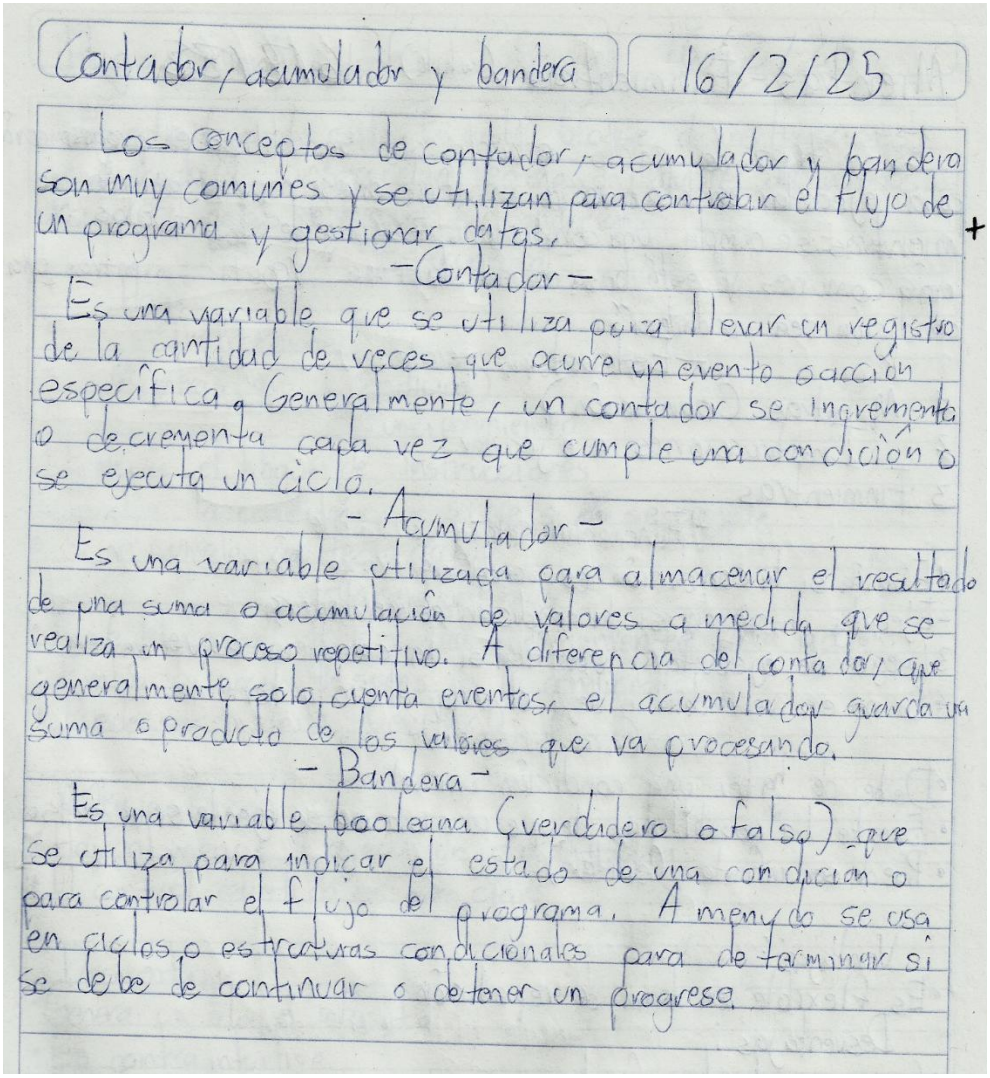


Ilustración 2. Representación de algoritmos con estructuras condicionales

En la imagen anterior se muestra una descripción simple de las 5 estructuras de control los cuales permiten la selección de decisiones, así como los diagramas de flujo respectivos de las siguientes estructuras condicionales: Simple, doble, múltiple, anidada y "switch".

Actividad 3



Contador, acumulador y bandera 16/2/25

Los conceptos de contador, acumulador y bandera son muy comunes y se utilizan para controlar el flujo de un programa y gestionar datos.

-Contador-

Es una variable que se utiliza para llevar un registro de la cantidad de veces que ocurre un evento o acción específica. Generalmente, un contador se incrementa o decrementa cada vez que cumple una condición o se ejecuta un ciclo.

- Acumulador -

Es una variable utilizada para almacenar el resultado de una suma o acumulación de valores a medida que se realiza un proceso repetitivo. A diferencia del contador, que generalmente solo cuenta eventos, el acumulador guarda la suma o producto de los valores que va procesando.

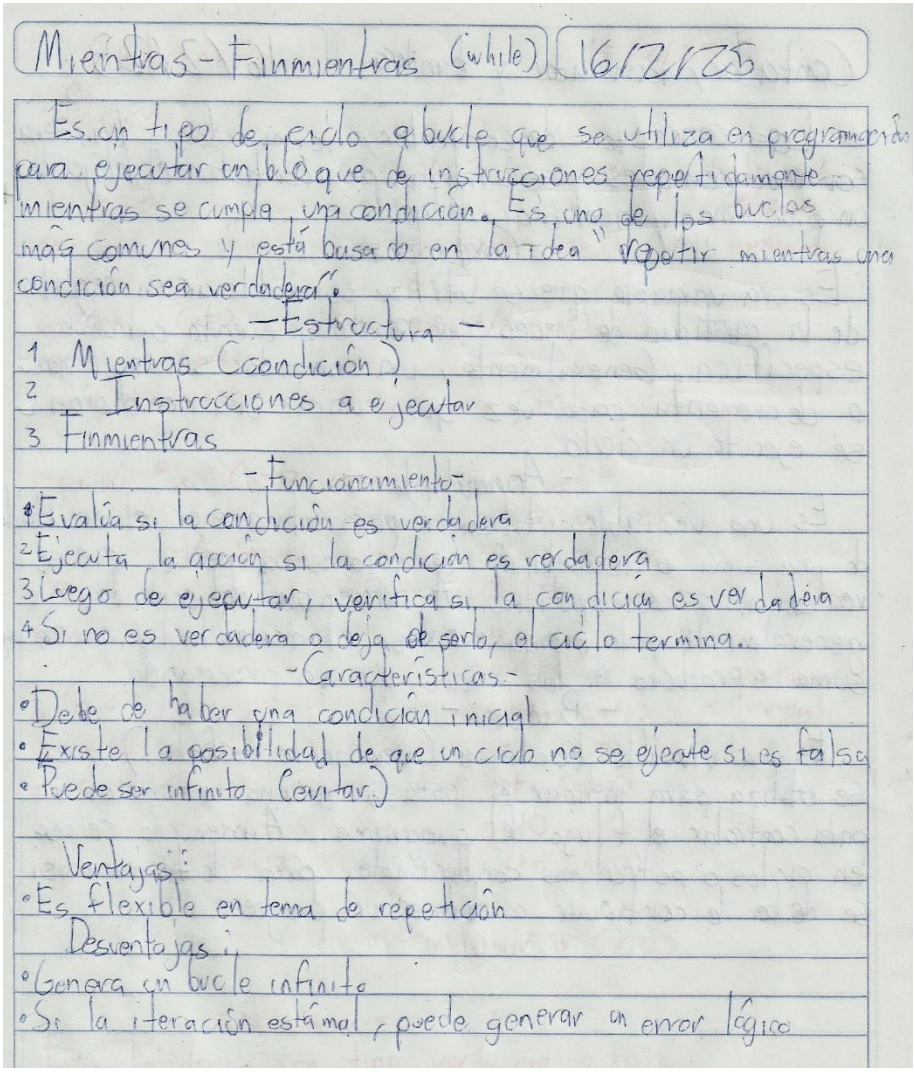
- Bandera -

Es una variable booleana (verdadero o falso) que se utiliza para indicar el estado de una condición o para controlar el flujo del programa. A menudo se usa en ciclos o estructuras condicionales para determinar si se debe de continuar o detener un progreso.

Ilustración 3. Contador, acumulador y bandera

En la imagen anterior se puede mostrar una descripción sencilla del contador, del acumulador y de la bandera, junto a su funcionamiento directo de cada uno de ellos y sus diferencias en caso del contador y del acumulador.

Actividad 4


Mientras - Finmientras (while) 16/2/25

Es un tipo de ciclo o bucle que se utiliza en programación para ejecutar un bloque de instrucciones repetidamente mientras se cumpla una condición. Es una de los bucles más comunes y está basado en la idea "repetir mientras una condición sea verdadera".

-Estructura-

1 Mientras (condición)
2 Instrucciones a ejecutar
3 Finmientras

-Funcionamiento-

1 Evalúa si la condición es verdadera
2 Ejecuta la acción si la condición es verdadera
3 Luego de ejecutar, verifica si la condición es verdadera
4 Si no es verdadera o deja de serlo, el ciclo termina.

-Características-

- Debe de haber una condición inicial
- Existe la posibilidad de que un ciclo no se ejecute si es falsa
- Puede ser infinito (evitar)

Ventajas:

- Es flexible en tema de repetición

Desventajas:

- Genera un bucle infinito
- Si la iteración está mal, puede generar un error lógico

Ilustración 4. Mientras – Fin mientras (while)

En la imagen anterior se muestra las características de la función mientras – fin mientras, también conocido como “while”. A su vez, se puede ver la estructura, el funcionamiento, las características y sus ventajas y desventajas.

Actividad 5

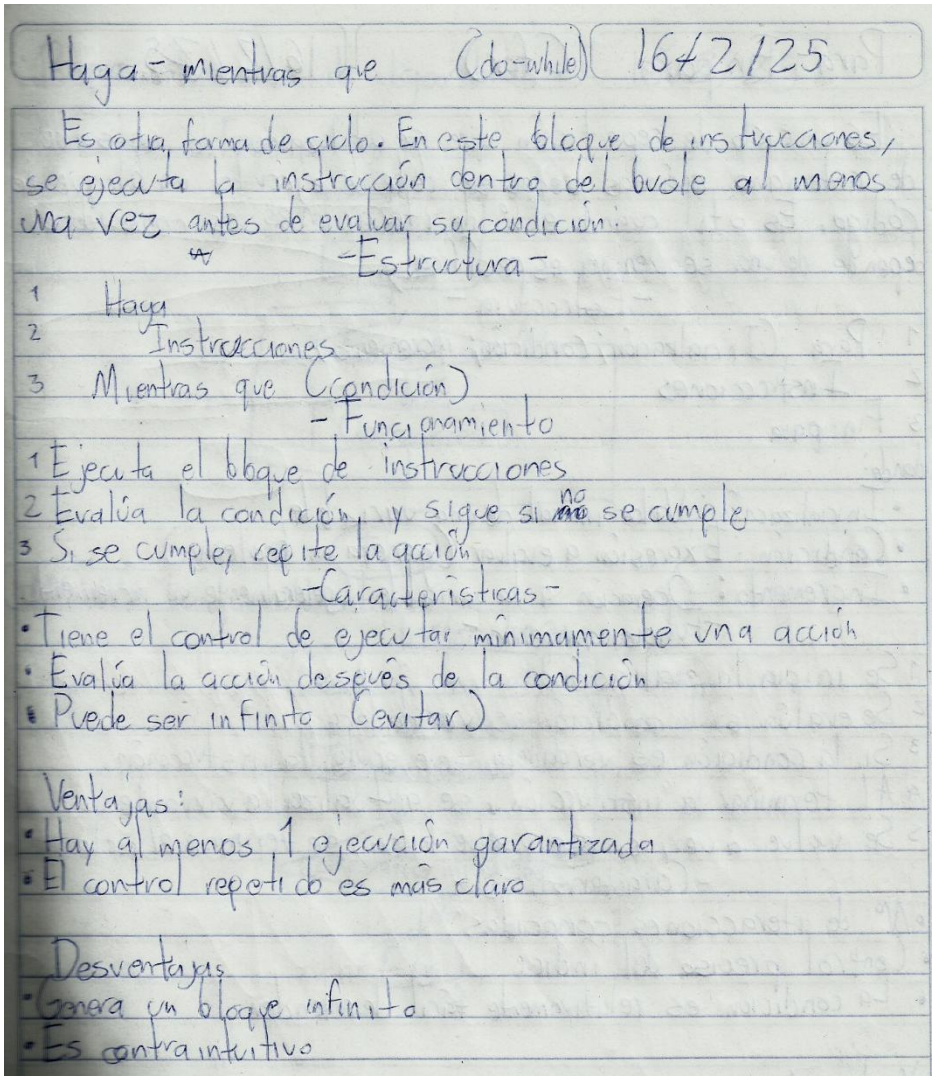
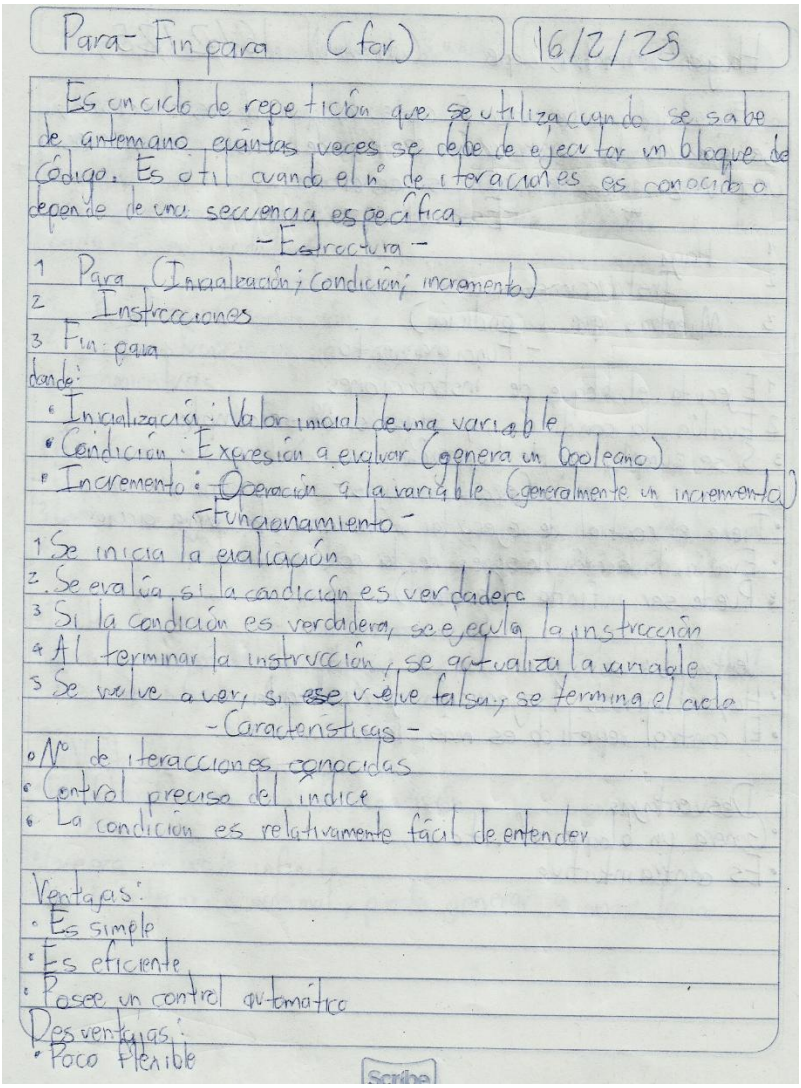


Ilustración 5. Haga - Mientras que (do - while)

En la imagen anterior se muestra las características de la función haga – mientras que, también conocido como “do - while”. A su vez, se puede ver la estructura, el funcionamiento, las características y sus ventajas y desventajas.

Actividad 6



Para - Fin para (for) 16/2/23

Es un ciclo de repetición que se utiliza cuando se sabe de antemano cuántas veces se debe de ejecutar un bloque de código. Es útil cuando el nº de iteraciones es conocido o depende de una secuencia específica.

-Estructura-

1 Para (Inicialización; Condición; incremento)
2 Instrucciones
3 Fin-para

donde:

- Inicialización: Valor inicial de una variable
- Condición: Expresión a evaluar (genera un booleano)
- Incremento: Operación a la variable (generalmente un incremento)

-Funcionamiento-

1. Se inicia la evaluación
2. Se evalúa si la condición es verdadera
3. Si la condición es verdadera, se ejecuta la instrucción
4. Al terminar la instrucción, se actualiza la variable
5. Se vuelve a ver, si ese vuelve falso, se termina el ciclo

-Características-

- Nº de iteraciones conocidas
- Control preciso del índice
- La condición es relativamente fácil de entender

Ventajas:

- Es simple
- Es eficiente
- Posee un control automático

Desventajas:

- Poco flexible

Scribe

Ilustración 6. Para - Fin para (for)

En la imagen anterior se muestra las características de la función para – fin para, también conocido como “for”. A su vez, se puede ver la estructura, el funcionamiento, las características y sus ventajas y desventajas.

Actividad 7

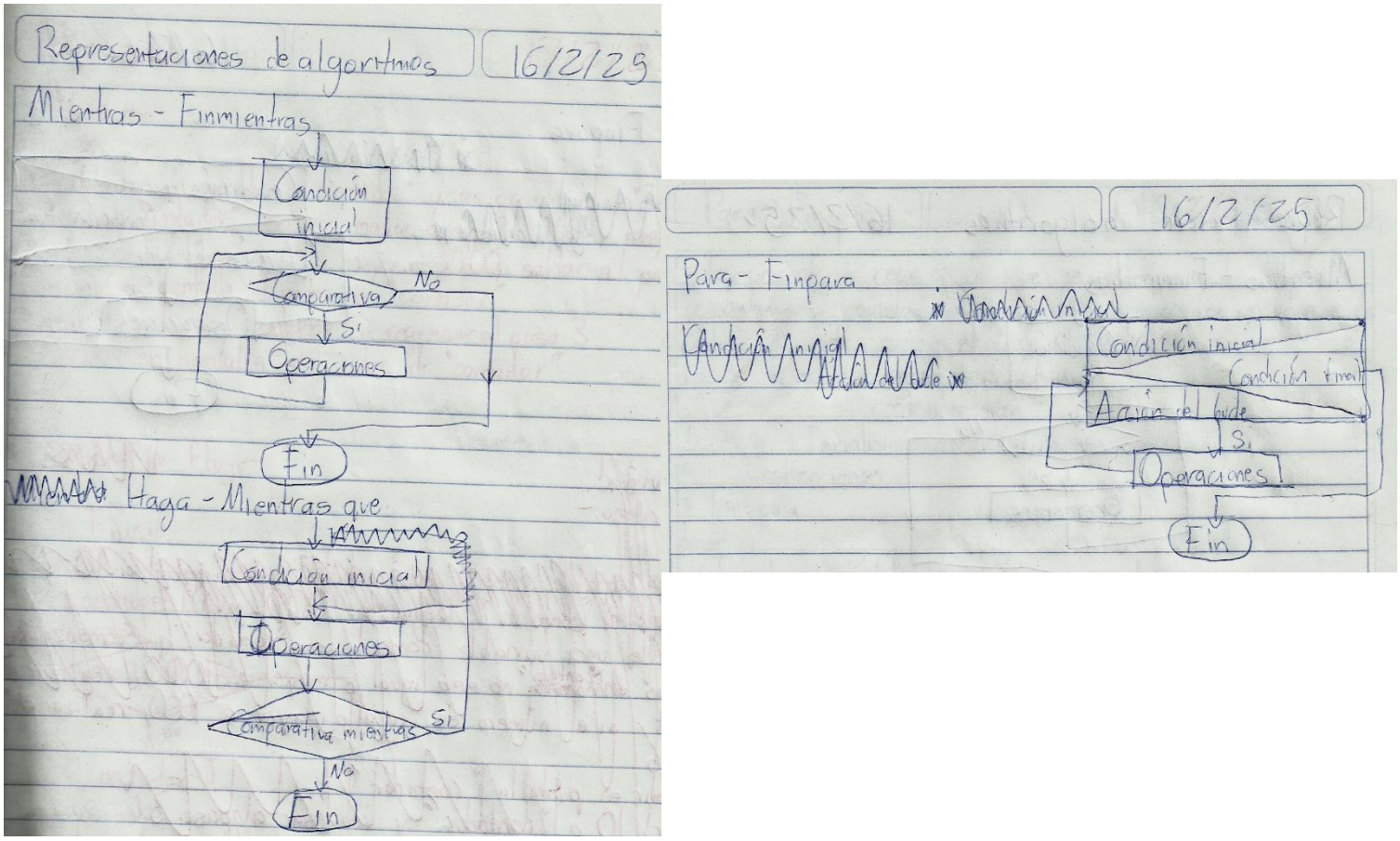


Ilustración 7. Representación de algoritmos

En esta actividad se puede observar la representación de los algoritmos Mientras – Fin mientras (while), Haga – Mientras que (do – while) y el Para – Fin para (for) gráficamente mediante las reglas utilizadas en un diagrama de flujo.

Retroalimentación

Estas actividades nos permitieron visualizar aquel tipo de algoritmo no secuencial el cual no fue descrito en la unidad anterior, siendo aquellos que se separan en múltiples ramas con acciones distintas cada uno o aquellos que regresan tras que una condición sea o no sea dada.

Este tipo de ejecuciones poseen mucha más utilidad que los algoritmos secuenciales presentados anteriormente podrían llegar a tener, permitiendo la fragmentación y desfragmentación de la ruta con propósitos de control y de repetición con la capacidad de resumir.

Esta actividad me ha permitido expandir mi razonamiento con la programación mediante la creación de rutas diferentes en caso de que ciertas condiciones se cumplan, mostrándome la capacidad de decisiones como un fundamento de la programación