Universidad Corporativa | Programas Académicos

Ingeniería en Desarrollo de Software

Proyecto # 2 | Diagrama De Flujos

Introducción al desarrollo de software

Ingeniería en Desarrollo de Software

academia global

TUTOR: SANDRA LARA DEVORA

ALUMNO: JOSE JOEL LANDEROS SANTOS

FECHA: 1 DICIEMBRE DE 2024

Introducción

En este ejercicio se aprenderá sobre los diagramas de flujo, una herramienta esencial para la representación visual de procesos y algoritmos. Los diagramas de flujo permiten organizar y estructurar las tareas a realizar de forma clara y secuencial, facilitando la comprensión de un proceso y su ejecución. A través de esta representación gráfica, se logra una visión más intuitiva de cómo los diferentes pasos de un programa o procedimiento interactúan entre sí, asegurando que cada acción sea comprensible y lógica.

En este proyecto, se explorará cómo los diagramas de flujo pueden descomponer procesos complejos en pasos más simples, lo que facilita su análisis y mejora la toma de decisiones. Además, se aprenderá cómo utilizar estos diagramas para identificar posibles errores, optimizar recursos y garantizar que el flujo de ejecución sea eficiente y claro. Esta herramienta es fundamental para el diseño de sistemas y la documentación de procesos, proporcionando una base sólida para desarrollar soluciones funcionales y efectivas.

Descripción

Un diagrama de flujo es una representación gráfica de un proceso, que utiliza símbolos y flechas para mostrar de manera secuencial los pasos necesarios para alcanzar un objetivo o resolver un problema. Cada símbolo tiene un significado específico, como indicar decisiones, procesos o entradas y salidas, lo que facilita la comprensión del flujo de actividades. Esta herramienta permite descomponer tareas complejas en pasos simples, haciendo que el proceso sea más claro y accesible tanto para expertos como para personas con menos experiencia en el tema.

En la programación y otros campos, los diagramas de flujo son fundamentales para visualizar y planificar procesos antes de implementarlos. Ayudan a identificar posibles errores o ineficiencias, optimizando el desarrollo de soluciones y asegurando que cada paso del proceso sea lógico y necesario. Su capacidad para transformar ideas abstractas en representaciones concretas los convierte en una herramienta indispensable en el diseño, documentación y análisis de sistemas y proyectos.

Justificación

PSInt es una excelente opción para principiantes, ya que permite plasmar algoritmos de manera sencilla y visual, utilizando un pseudocódigo intuitivo que ayuda a entender la estructura lógica de los programas. Al trabajar en PSInt, los usuarios pueden practicar el diseño de algoritmos sin preocuparse por la sintaxis estricta de un lenguaje de programación, lo que refuerza la lógica antes de dar el siguiente paso hacia la codificación. Además de que ofrece una representación clara y estructurada del flujo de un programa. Esto no solo es útil para el aprendizaje, sino también para proyectos colaborativos o laborales, donde la comunicación de ideas debe ser precisa y accesible para todos los involucrados. Al integrar herramientas como PSInt en la práctica diaria, los desarrolladores principiantes adquieren una base sólida para abordar problemas más complejos en el futuro.

Desarrollo

Calculadora: Primos

La empresa MathTech requiere a un ingeniero en desarrollo de software que sea capaz de realizar la tarea de programar tres tipos de calculadoras diferentes para implementar en los colegios y escuelas públicas: - La primera calculadora deberá de llevar por nombre Primos, y su objetivo será identificar los números primos que se ingresen, por ejemplo si el usuario ingresa el número 83, deberá imprimir el siguiente mensaje: “El número (número ingresado) si es primo”, en caso de que no sea primo se imprimirá el siguiente mensaje “El número (número ingresado) no es primo”.

Algoritmo:

1. Ingresar un número.
2. Crear una variable booleana que nos indique si es primo o no.
3. Si el número es menor o igual a 1, establecer la variable booleana como falso.
4. Recorrer los números desde 2 hasta la raíz cuadrada del número ingresado:
    - Si el número ingresado es divisible por algún número del rango, establecer el booleano como verdadero.
5. Si el booleano es verdadero, imprimir “El número (número ingresado) si es primo”. De lo contrario, imprimir “El número (número ingresado) no es primo”.

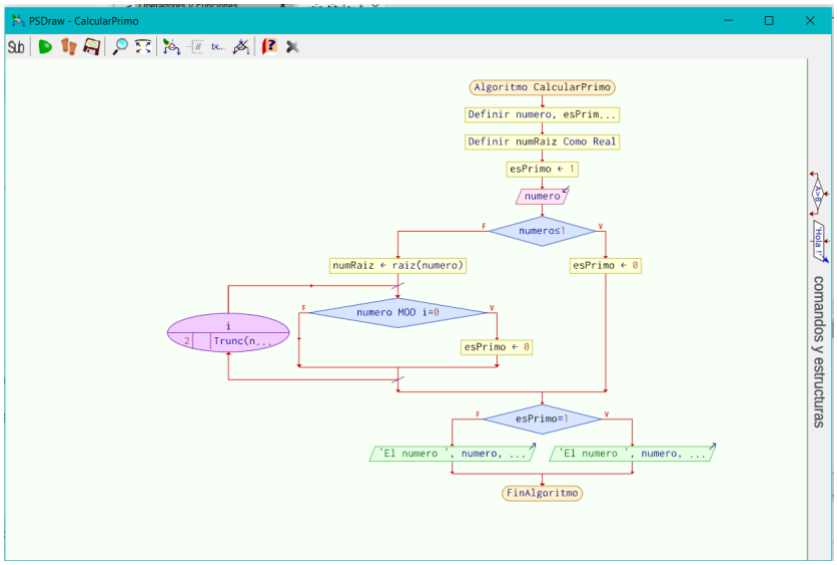

Imagen 1.1 captura de pantalla de PSint donde muestra el diagrama de flujo en base al seudocódigo

Tomando en cuenta el diagrama de flujo mostrado en la Imagen 1.1, podemos observar cómo se estructura claramente la ruta que toma el código, permitiéndonos comprender su funcionamiento de manera visual. El rombo azul representa una decisión lógica basada en un valor booleano, dividiendo el flujo en dos posibles rutas. Esto facilita identificar las condiciones que se deben evaluar y las acciones que se deben realizar dependiendo del resultado.

Calculadora: Par/Impar

La segunda calculadora se llamará Par/Impar, su objetivo es que se ingresen 10 números, ya sean pares o impares, por ejemplo, si se ingresa el número 9, el programa deberá de indicar que es un número impar, pero si se trata del número 2, el programa deberá indicar que se trata de un número par. De 10 números enteros, se debe determinar cuáles son pares y cuáles son impares.

Algoritmo:

1. Ingresar 10 números enteros.
2. Para cada número:
    - Verificar con un condicional si es divisible entre 2:
        - Resultado arroja verdadero: Es par.
        - Resultado arroja falso: Es impar.
3. Imprimir el resultado de cada número.

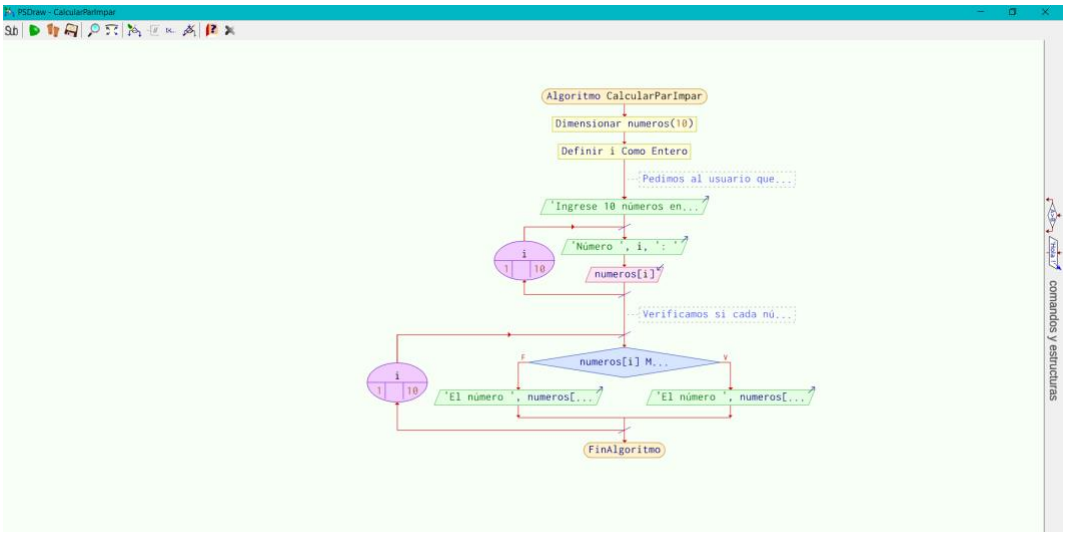

Imagen 1.2 captura de pantalla de PSint donde muestra el diagrama de flujo en base al seudocódigo

A pesar de que el flujo representado en el diagrama es relativamente sencillo, uno de los aspectos que puede present mayor complejidad es la definición y manipulación de arreglos. Este es un punto que siempre requiere una atención especial y, en muchos casos, se debe consultar la documentación para garantizar su correcta implementación. En el diagrama, el círculo rosado destaca que una acción se repetirá para cada uno de los 10 elementos del arreglo, lo cual ilustra claramente el uso de ciclos y cómo se gestionan los datos de manera iterativa.

Calculadora: Al Revés

El último programa se llamará Al Revés, su objetivo es que el usuario ingrese un número de 4 dígitos y que sea un número entero, y este programa se encargará de regresar los números al revés o invertidos. Por ejemplo, si se ingresa el número 7631, el programa matemático deberá regresar 1367.

Algoritmo:

1. Ingresar un número entero de 4 dígitos.
2. Confirmar que el número tenga 4 dígitos.
3. Separar los dígitos y guardarlos en esa secuencia por caracter
4. Crear otro número en orden inverso usando una iteracion de cadena de caracteres.
5. Imprimir el número invertido.

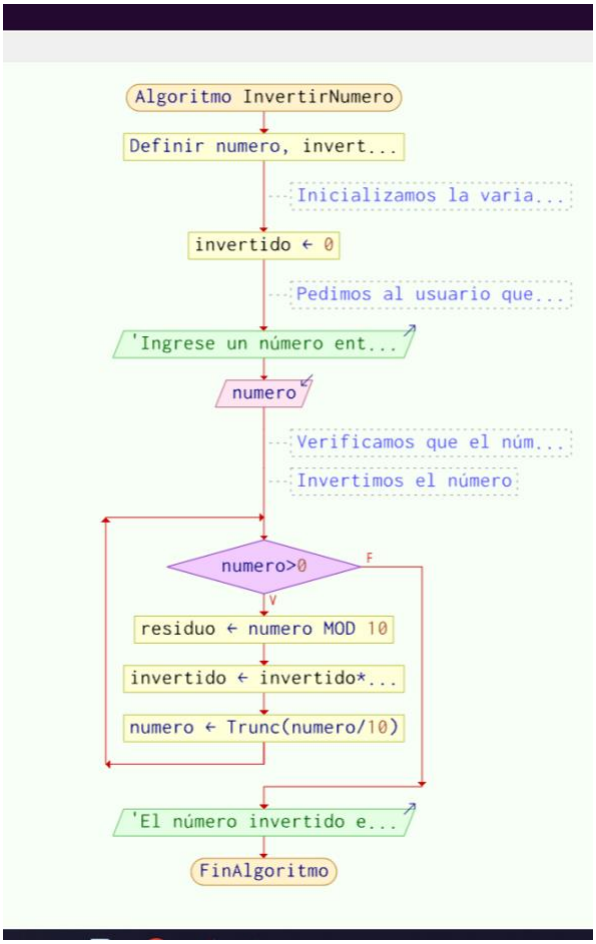

Imagen 1.3 captura de pantalla de PSint donde muestra el diagrama de flujo en base al seudocódigo.

Al utilizar el residuo de una división entre 10, obtienes el último dígito del número, lo que te permite invertir la secuencia de los dígitos uno a uno. Al multiplicar el número invertido por 10 en cada iteración y sumarle el residuo, consigues revertir el orden de los dígitos.

Conclusión

La práctica con diagramas de flujo y algoritmos me permitió comprender la importancia de descomponer problemas en pasos claros y secuenciales. Este enfoque facilita tanto la planeación como la implementación de soluciones, asegurando que cada acción dentro de un programa sea lógica y cumpla con su propósito. A través de estas actividades, desarrollé un pensamiento crítico y analítico que me ayudó a identificar posibles errores y áreas de mejora antes de llevar los procesos al código, optimizando tiempo y esfuerzo en futuras actualizaciones o modificaciones.

Entendí también que los diagramas de flujo son herramientas fundamentales para visualizar y organizar la lógica detrás de los algoritmos. Herramientas como PSInt complementaron este aprendizaje al permitirme plasmar de manera sencilla y clara los pasos necesarios para resolver un problema. Este proceso no solo fortaleció mi capacidad para diseñar soluciones estructuradas, sino que también me proporcionó una base sólida para enfrentar desafíos más complejos en el ámbito de la programación y la resolución de problemas computacionales.

Referencias

Francisco Javier Pinales Delgado. (2014). ROBLEMARIO DE ALGORITMOS RESUELTOS CON DIAGRAMAS [Libro] Universidad Autónoma de Aguascalientes.