Instituto Politécnico Nacional

Escuela Superior de Computo

Fundamentos De Programación

Pérez García Edgar Israel.

Entrada y salida en C.

30 de septiembre de 2025.

Introducción.

En su forma más fundamental, un programa en C ejecuta sus instrucciones de manera secuencial, una tras otra, desde el inicio de la función main hasta su final. Este orden predecible, similar a seguir una receta paso a paso, es la base de la computación. Sin embargo, para crear aplicaciones que sean verdaderamente útiles y no meros autómatas, es imprescindible contar con la capacidad de alterar este flujo lineal. El **control de flujo** es precisamente el conjunto de mecanismos que ofrece C para romper la secuencia, permitiendo al programa tomar decisiones, reaccionar a diferentes entradas y ejecutar tareas de forma repetitiva. Estas estructuras son las que dotan al software de lógica, flexibilidad y dinamismo.

La primera gran categoría de control de flujo es la **selección o bifurcación condicional**, que permite al programa elegir entre dos o más rutas de ejecución. El pilar de esta lógica es la sentencia **if**, la cual evalúa una expresión booleana (una condición que resulta ser verdadera o falsa) y ejecuta un bloque de código solo si el resultado es verdadero. Para manejar el caso contrario, if se complementa con **else**, que provee un camino alternativo a ejecutar si la condición es falsa. Cuando existen múltiples condiciones a evaluar, la estructura **else if** permite encadenarlas de manera ordenada y excluyente. Para casos más específicos donde las decisiones dependen del valor concreto de una única variable entera o de tipo carácter, C provee la estructura **switch**. Esta sentencia actúa como un despachador que dirige el flujo a uno de varios bloques de código (case) según el valor de la variable, ofreciendo una alternativa más legible y a menudo más eficiente que una larga serie de if-else.

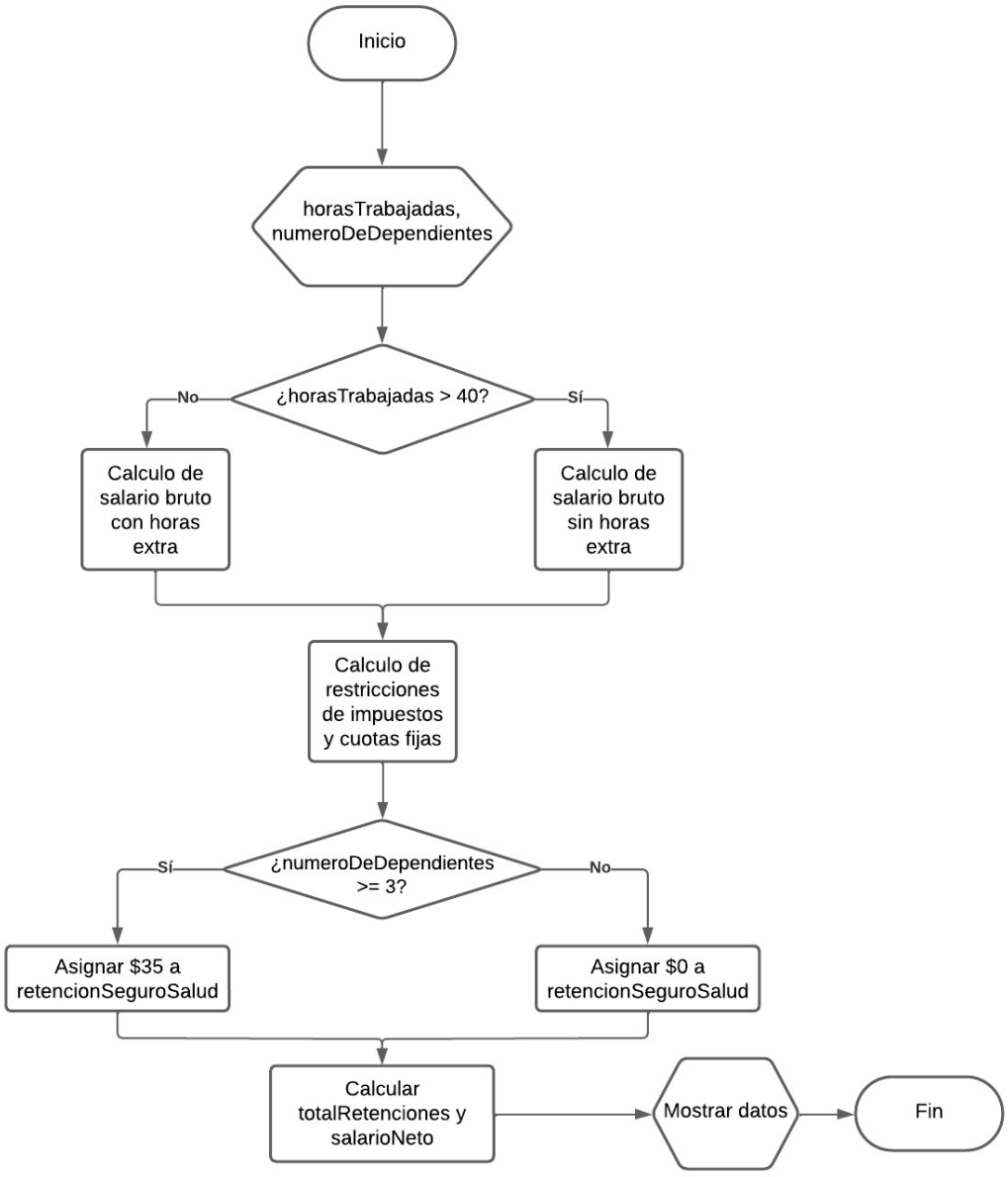
La segunda categoría fundamental es la **repetición o iteración**, diseñada para ejecutar un bloque de código múltiples veces. Todo bucle se fundamenta en tres acciones clave: una inicialización de variables de control, una condición que determina si el bucle continúa, y una actualización de dichas variables en cada ciclo. El bucle **for** es la estructura iterativa por excelencia cuando se conoce de antemano el número de repeticiones, ya que encapsula estos tres componentes (inicialización, condición y actualización) en una sola línea de manera clara y concisa. Por otro lado, el bucle **while** repite una sección de código mientras una condición permanezca verdadera, evaluando

dicha condición antes de cada ejecución. Su flexibilidad radica en que la actualización de la variable de control debe manejarse explícitamente dentro del cuerpo del bucle. La omisión de este paso es una causa común de los temidos bucles infinitos. Una variante, el bucle **do-while**, garantiza que el código se ejecute al menos una vez, ya que su condición se evalúa al final de cada iteración.

Adicionalmente, C ofrece **sentencias de salto** que modifican el comportamiento de estas estructuras. La sentencia **break** permite una salida abrupta de un bucle o de una estructura switch, mientras que **continue** omite el resto del código de la iteración actual y salta directamente al inicio de la siguiente. El dominio combinado de las estructuras condicionales, los bucles y las sentencias de salto es, por tanto, un pilar esencial en la programación, pues son las herramientas que permiten trascender un guion rígido para construir un programa inteligente y adaptable.

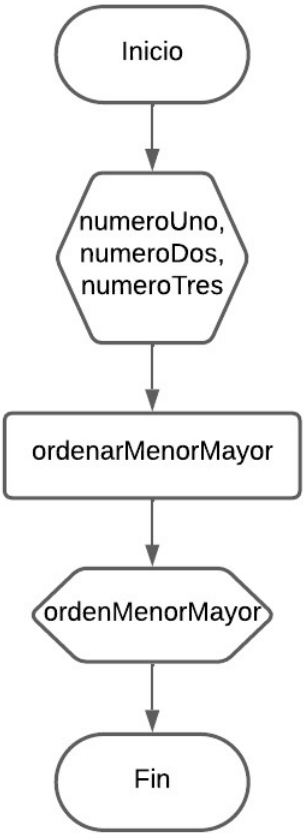
Desarrollo.

1. A un empleado se le paga a razón de $16.78 por hora durante las primeras 40 horas trabajado en una semana. Cualquier hora extra que se pague a la tasa de horas extras de uno y la mitad de eso. Del salario bruto del trabajador se retiene el 6% para Impuesto del Seguro Social, se retiene el 14 % para el impuesto federal sobre la renta, se retiene el 5 % para el impuesto estatal sobre la renta, y se retienen $10 por semana para las cuotas sindicales. Si el trabajador tiene tres o más dependientes, entonces se retienen $35adicionales para cubrir el costo adicional del seguro de salud más allá de lo que paga el empleador. Escriba un programa que lea el número de horas trabajadas en una semana y el número de dependientes como entrada y luego generará la salida del trabajador salario bruto, cada monto de retención y el salario neto por semana.



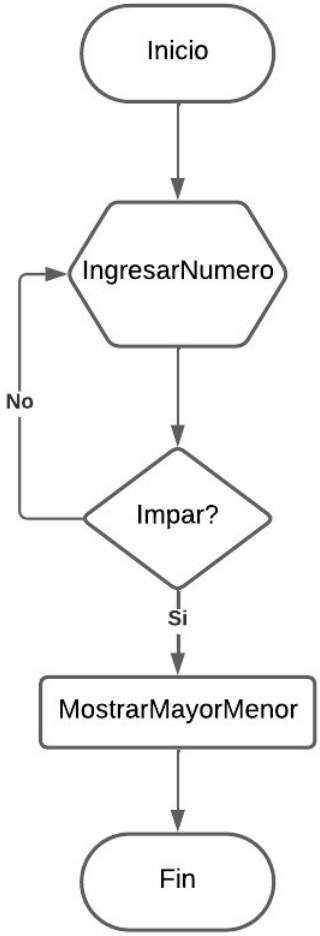
Se desarrolló el código fuente en PracticaCuatroSalario.c contenido en la misma carpeta que este documento

2. Escriba un programa que lea tres valores enteros. Entonces los números deberían salir en orden ascendente de menor a mayor. Puedes hacer esto solo con sentencias if y tres variables de tipo int. Sugerencia: intente anidar sentencias if o usando el operador &&, ¿Qué sucede si ingresa tres números idénticos?



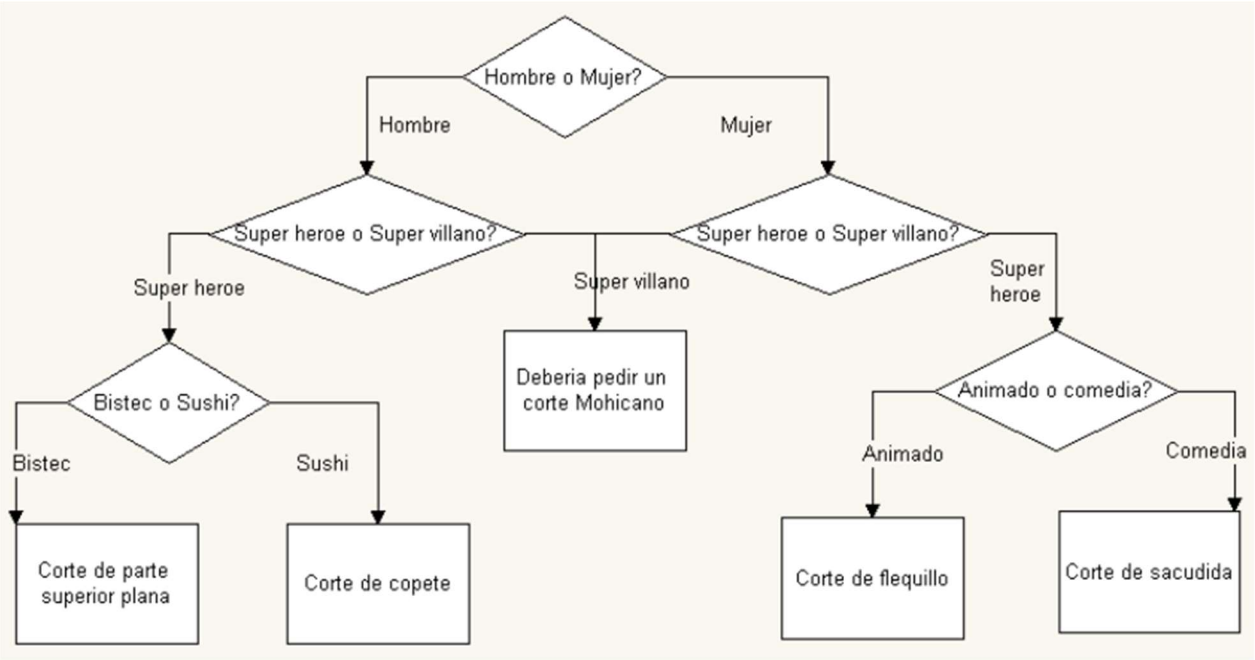
Se desarrolló el código fuente en PracticaCuatroMenoraMayor.c contenido en la misma carpeta que este documento

3. Escriba un programa que lea valores de números enteros (tipo int) del usuario hasta que ingrese un número negativo, por ejemplo -21. Una vez que el usuario haya terminado de ingresar números, imprimir el valor más alto que haya ingresado, el valor más bajo que haya ingresado, y el número total de números que ingresó. El número negativo que se ingresa no debe tomarse como uno de los valores, solo es para detener la lectura de números del usuario.



Se desarrolló el código fuente en PracticaCuatroValorAlto.c contenido en la misma carpeta que este documento

4. El siguiente diagrama de flujo contiene una serie de preguntas para determinar qué tipo de corte de pelo ha de conseguir. Escribir un programa que haga las preguntas al usuario y genere el corte de cabello recomendado.



Se desarrolló el código fuente en PracticaCuatroHeroeVillano.c contenido en la misma carpeta que este documento

Conclusión

Esta práctica ha sido fundamental para consolidar la comprensión teórica y la aplicación práctica de las **estructuras de control de flujo** en el lenguaje de programación C. A través de la resolución de los cuatro problemas planteados, se logró trascender la ejecución secuencial de instrucciones para implementar programas con lógica, flexibilidad y capacidad de decisión.

El desarrollo de los ejercicios permitió materializar los conceptos presentados en la introducción. El primer problema, sobre el cálculo de salarios, demostró la utilidad de las sentencias condicionales if-else para manejar casos de negocio variables, como el pago de horas extra y las deducciones basadas en el número de dependientes. El segundo y cuarto ejercicio, que consistían en ordenar números y crear un recomendador de cortes de cabello, evidenciaron cómo las estructuras if anidadas son capaces de traducir diagramas de flujo complejos y árboles de decisión en código funcional y legible.

Finalmente, el tercer problema, que requería encontrar el valor más alto y más bajo de una serie de números, fue una excelente aplicación de los **bucles iterativos**. Este ejercicio demostró cómo un ciclo do-while o while, combinado con una condición de

término (un valor centinela), es una herramienta poderosa para procesar una cantidad indeterminada de datos ingresados por el usuario.

En resumen, la práctica confirma que el dominio de las estructuras condicionales (como if y switch) y los bucles (for, while, do-while) es un pilar esencial en la programación. Estas herramientas son las que permiten que un programa deje de ser un guion rígido para convertirse en una solución inteligente y adaptable, capaz de reaccionar a diferentes entradas y ejecutar tareas complejas de manera eficiente.

Referencias

Deitel, P. J., & Deitel, H. M. (2016). *C How to Program* (8th ed.). Pearson.

Kernighan, B. W., & Ritchie, D. M. (2014). *The C Programming Language* (2nd ed.). Pearson Education.

Kochan, S. G. (2014). *Programming in C* (4th ed.). Addison-Wesley Professional.

Prata, S. (2013). *C Primer Plus* (6th ed.). Addison-Wesley Professional.