Diagramas de flujo, pseudocódigo y pruebas de escritorio.

Objetivo:

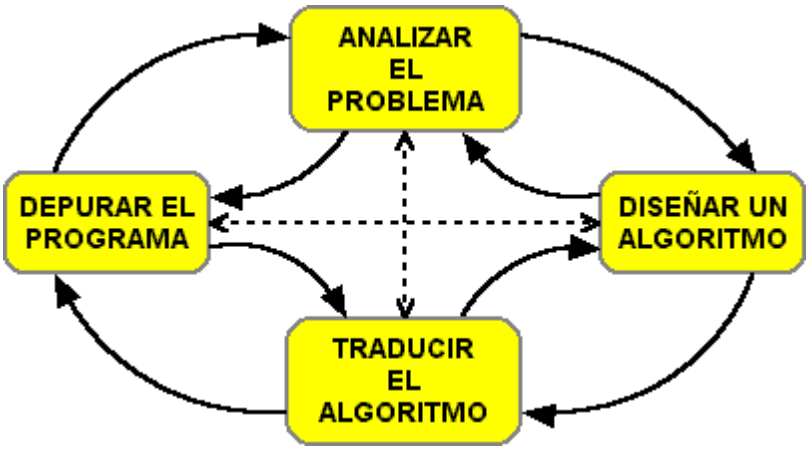
Al terminar la instrucción, el alumno tendrá la capacidad de actuar creativamente para elaborar programas que resuelvan situaciones planteadas por el instructor.

Desarrollo:

Algoritmo:

Algoritmo: Un algoritmo es una lista de instrucciones bien definida, ordenada y finita mediante las cuales se efectuará paso a paso un proceso para obtener un cierto resultado, un ejemplo de ello son las recetas de cocina, manuales, etc.

Los programas de computadora tienen como finalidad resolver problemas específicos y el primer paso consiste en definir con precisión el problema hasta lograr la mejor comprensión posible. Una forma de realizar esta actividad se basa en formular claramente el problema, especificar los resultados que se desean obtener, identificar la información disponible (datos), determinar las restricciones y definir los procesos necesarios para convertir los datos disponibles (materia prima) en la información requerida (resultados).



Fases para elaborar un programa

Se pueden identificar las siguientes sugerencias propuestas a los estudiantes para llegar a la solución de un problema matemático:

1. COMPRENDER EL PROBLEMA.
    - Leer el problema varias veces
    - Establecer los datos del problema
    - Aclarar lo que se va a resolver (¿Cuál es la pregunta?)
    - Precisar el resultado que se desea lograr
    - Determinar la incógnita del problema
    - Organizar la información
    - Agrupar los datos en categorías
    - Trazar una figura o diagrama.
2. HACER EL PLAN.
    - Escoger y decidir las operaciones a efectuar.
    - Eliminar los datos inútiles.
    - Descomponer el problema en otros más pequeños.
3. EJECUTAR EL PLAN (Resolver).
    - Ejecutar en detalle cada operación.
    - Simplificar antes de calcular.
    - Realizar un dibujo o diagrama
4. ANALIZAR LA SOLUCIÓN (Revisar).
    - Dar una respuesta completa
    - Hallar el mismo resultado de otra manera.
    - Verificar por apreciación que la respuesta es adecuada.

EJEMPLO

En un juego, el ganador obtiene una ficha roja; el segundo, una ficha azul; y el tercero, una amarilla. Al final de varias rondas, el puntaje se calcula de la siguiente manera: Al cubo de la cantidad de fichas rojas se adiciona el doble de fichas azules y se descuenta el cuadrado de las fichas amarillas. Si Andrés llegó 3 veces en primer lugar, 4 veces de último y 6 veces de intermedio, ¿Qué puntaje obtuvo?

R/.

COMPRENDE
- Leer detenidamente el problema
- ¿Cuántos colores de fichas se reparten?
- ¿Cuántas fichas rojas, azules y amarillas obtuvo Andrés?
- ¿Qué pregunta el problema?

PLANEA
- Para hallar el puntaje que obtiene Andrés por sus llegadas de primero, calcular el cubo de la cantidad de fichas rojas.
- Para hallar el puntaje por sus llegadas en segundo lugar, calcular el doble de la cantidad de fichas azules.
- Para hallar el puntaje que pierde por sus llegadas en último lugar, calcular el cuadrado de la cantidad de fichas amarillas.
- Para hallar el puntaje total, calcular la suma de los puntajes por las fichas rojas y azules, restarle los puntos de las fichas amarillas.

RESUELVE
- Por tres fichas rojas: _____ puntos.
- Por seis fichas azules: _____ puntos.
- Por cuatro fichas amarillas: _____ puntos.

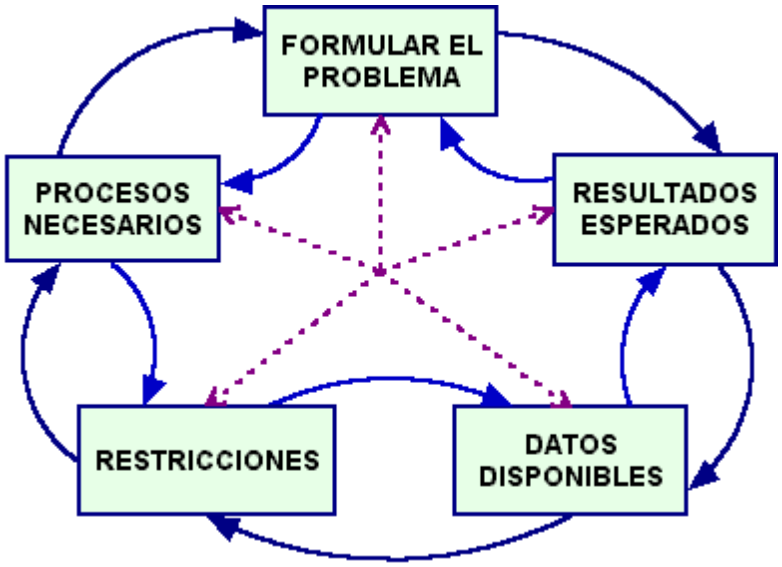
• Para obtener el puntaje final de Andrés, sumar los puntos obtenidos con las fichas rojas y azules (_____puntos) y de este resultado restar los puntos representados por las fichas amarillas (_____puntos).

REVISA

• El puntaje que obtuvo Andrés es ___ puntos.

• Verificar las operaciones y comparar los cálculos con la solución estimada.

Analizar el problema (entenderlo)



Etapas a desarrollar en la fase de análisis

Como ya se mencionó al inicio del capítulo lo primero a realizar es definir el problema con precisión para comprenderlo de una manera mejor.
Estas etapas coinciden parcialmente con los elementos generales que están presentes en todos los problemas:

1. Especificar claramente los resultados que se desean obtener (meta y sub-metas)
2. Identificar la información disponible (estado inicial)
3. Definir los procesos que llevan desde los datos disponibles hasta el resultado deseado (operaciones)

Para establecer un modelo que los estudiantes puedan utilizar en la fase de análisis del problema, debemos agregar dos temas a los elementos expuestos: formular el problema y determinar las restricciones.

Formular el problema

La solución de un problema debe iniciar por determinar y comprender exactamente en qué consiste ese problema. Esta etapa es una buena oportunidad para plantear situaciones en forma verbal o escrita que vinculen la enseñanza de las matemáticas con el entorno en el que vive el estudiante y que tengan una variedad de estructuras y de formas de solución.

Esta metodología obliga al estudiante a formular el problema a partir de la situación real planteada. De esta manera se contrarresta la costumbre tan común en el aula de que los problemas sean formulados por el profesor o tomados de los libros de texto.

EJEMPLO
OPCIÓN 1:
Juan Felipe es jefe de bodega en una fábrica de pañales desechables y sabe que la producción diaria es de 744 pañales y que en cada caja donde se empacan para la venta caben 12 pañales. ¿Cuántas cajas debe conseguir Juan Felipe para empacar los pañales fabricados en una semana?

OPCIÓN 2:
Juan Felipe es jefe de bodega en una fábrica de pañales desechables y una de las tares del día consiste en llamar al proveedor de los empaques y ordenarle la cantidad suficiente de cajas para empacar los pañales fabricados en la semana próxima. El jefe de producción le informó ayer a Juan Felipe que la producción diaria será de 744 pañales y en cada caja cabe una docena de ellos.
¿Qué debe hacer Felipe?

La Opción 1 plantea directamente el problema que el estudiante debe resolver. Mientras que la Opción 2 plantea una situación y la pregunta es ¿Qué debe hacer Felipe?. La Opción 2 demanda al estudiante leer muy bien el texto para comprender la situación y así poder formular el problema de Juan Felipe. Es algo similar a preguntar al estudiante "cuánto es 7 menos 3" versus preguntar "sí Rosa tiene 7 naranjas y Julio tiene 3, cuántas naranjas de más tiene Rosa".

La comprensión lingüística del problema (entender el significado de cada enunciado) es muy importante. El estudiante debe realizar una lectura previa del problema con el fin de obtener una visión general de lo que se le pide y una segunda lectura para poder responder a preguntas como:

- ¿Puedo definir mejor el problema?
- ¿Qué palabras del problema me son desconocidas?
- ¿Cuáles son las palabras clave del problema?
- ¿He resuelto antes algún problema similar?
- ¿Qué información es importante?
- ¿Qué información puedo omitir?

Además, es conveniente que los estudiantes se habitúen a analizar los problemas desde diferentes puntos de vista y a categorizar la información dispersa que reciben como materia prima. En programación es frecuente que quien programa deba formular el problema a partir de los resultados esperados. Es muy importante que el estudiante sea consciente de que cuando las especificaciones de un programa se comunican mediante lenguaje natural, estas pueden ser ambiguas, incompletas e incongruentes. En esta etapa se debe hacer una representación precisa del problema; especificar lo más exactamente posible lo que hay que hacer (no cómo hay que hacerlo).

EJEMPLO

Doña Rubí necesita decidir cómo comprar un televisor que cuesta 850.000 de contado o 960.000 a crédito. Ella tiene 600.000 pesos en efectivo.
R/.
Como el efectivo que tiene doña Rubí no le alcanza para comprar el televisor de contado, ella tiene dos opciones: comprarlo totalmente a crédito o pagar una parte de contado (cuota inicial) y el resto a crédito.
Para poder resolver el problema se debe conocer el número de cuotas si desea pagarlo totalmente a crédito o conocer el número de cuotas y el valor total del televisor si se da una cuota inicial de 600.000 pesos.

Precisar los resultados esperados (meta y sub-metas)

Para establecer los resultados que se esperan (meta) es necesario identificar la información relevante, ignorar los detalles sin importancia, entender los elementos del problema y activar el esquema correcto que permita comprenderlo en su totalidad.
Determinar con claridad cuál es el resultado final (producto) que debe devolver el programa es algo que ayuda a establecer la meta. Es necesario analizar qué resultados se solicitan y qué formato deben tener esos resultados (impresos, en pantalla, diagramación, orden, etc.). El estudiante debe preguntarse:

- ¿Qué información me solicitan?
- ¿Qué formato debe tener esta información?

Identificar datos disponibles (estado inicial)

Otro aspecto muy importante en la etapa de análisis del problema consiste en determinar cuál es la información disponible. El estudiante debe preguntarse:

- ¿Qué información es importante?
- ¿Qué información no es relevante?
- ¿Cuáles son los datos de entrada? (conocidos)
- ¿Cuál es la incógnita?
- ¿Qué información me falta para resolver el problema? (datos desconocidos)
- ¿Puedo agrupar los datos en categorías?

Otro aspecto importante del estado inicial hace referencia al nivel de conocimiento que el estudiante posee en el ámbito del problema que está tratando de resolver. Es conveniente que el estudiante se pregunte a sí mismo:

- ¿Qué conocimientos tengo en el área o áreas del problema?
- ¿Son suficientes esos conocimientos?
- ¿Dónde puedo obtener el conocimiento que necesito para resolver el problema?
- ¿Mis compañeros de estudio me pueden ayudar a clarificar mis dudas?
- ¿Qué expertos en el tema puedo consultar?

En el ámbito de las matemáticas, se conoce como conocimiento condicional a aquel que activan los estudiantes cuando aplican procedimientos matemáticos concretos de manera intencional y consciente a ciertas situaciones. “El conocimiento condicional proporciona al alumno un sistema de valoración sobre la extensión y las limitaciones de su saber (qué sabe sobre el tema, su capacidad de memoria, etc.), a la vez que examina la naturaleza de la demanda del profesor y su objetivo último, y evalúa variables externas como pueden ser el tiempo que tiene o con quién realiza la tarea”.

EJEMPLO
Esteban está ahorrando para comprar una patineta que vale 55.000 pesos. Su papá le ha dado una mesada de 5.000 pesos durante 7 semanas. Por lavar el auto de su tío tres veces recibió 8.000 pesos. Su hermano ganó 10.000 pesos por hacer los mandados de su mamá y 4.000 por sacar a pasear el perro. ¿Esteban tiene ahorrado el dinero suficiente para comprar la patineta o aún le falta?
R/.
Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: Si o no tiene Esteban ahorrado el dinero suficiente para comprar una patineta que vale 55.000 pesos.
Datos disponibles: Los ingresos de Esteban: 5.000 pesos por 7 semanas + 8.000 pesos. Los 10.000 y 4.000 pesos qué ganó el hermano de Esteban son irrelevantes para la solución de este problema y se pueden omitir.

Determinar las restricciones

Resulta fundamental que los estudiantes determinen aquello que está permitido o prohibido hacer y/o utilizar para llegar a una solución. En este punto se deben exponer las necesidades y restricciones (no una propuesta de solución). El estudiante debe preguntarse:

- ¿Qué condiciones me plantea el problema?
- ¿Qué está prohibido hacer y/o utilizar?
- ¿Qué está permitido hacer y/o utilizar?
- ¿Cuáles datos puedo considerar fijos (constantes) para simplificar el problema?
- ¿Cuáles datos son variables?
- ¿Cuáles datos debo calcular?

Establecer procesos (operaciones)

Consiste en determinar los procesos que permiten llegar a los resultados esperados a partir de los datos disponibles. El estudiante debe preguntarse:

- ¿Qué procesos necesito?
- ¿Qué fórmulas debo emplear?
- ¿Cómo afectan las condiciones a los procesos?
- ¿Qué debo hacer?
- ¿Cuál es el orden de lo que debo hacer?

En la medida de lo posible, es aconsejable dividir el problema original en otros más pequeños y fáciles de solucionar (submetas), hasta que los pasos para alcanzarlas se puedan determinar con bastante precisión (módulos). Esto es lo que en programación se denomina diseño descendente o top-down.
El diseño descendente se utiliza en la programación estructurada de computadoras debido a que facilita:

- La comprensión del problema
- Las modificaciones en los módulos
- La verificación de la solución

Al realizar divisiones sucesivas del problema en otros más pequeños y manejables (módulos), hay que tener cuidado para no perder de vista la comprensión de este como un todo. El estudiante, luego de dividir el problema original en submetas (módulos), debe integrar cada parte de tal forma que le permita comprender el problema como un todo.

Igualmente hay que tener cuidado cuando se utiliza este enfoque para resolver problemas complejos o extensos, en cuyo caso resulta más aconsejable utilizar una metodología orientada a objetos. Especialmente, cuando profesores universitarios manifiestan su preocupación por el aprendizaje de malas prácticas de programación en el colegio. Hay casos en los cuales algunos estudiantes no han podido cambiar su forma de pensar "estructurada" por otra orientada a objetos, la cual hace parte de los programas universitarios modernos en la carrera de Ingeniería de Sistemas. Es aconsejable que los ejemplos y actividades planteados a los estudiantes contengan solo un problema cuya solución sea muy corta (no necesariamente sencillo de resolver).

De esta forma ellos podrán enfocarse en aplicar completamente la metodología propuesta para analizar problemas (formular el problema, especificar los resultados, identificar la información disponible, determinar las restricciones y definir los procesos) sin perderse en el laberinto de un problema demasiado complejo.

EJEMPLO
De acuerdo con la metodología descrita, analizar el problema de hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.
R/
Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: El área de un triángulo rectángulo.
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo. La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Determinar las restricciones: Utilizar las medidas dadas.
Procesos necesarios: Guardar en dos variables los valores de Base y Altura; Guardar en una constante el divisor 2; aplicar la fórmula área=base*altura/2; comunicar el resultado (área).

ACTIVIDAD
La mayoría de las metodologías propuestas para la solución de problemas matemáticos se aproxima al ciclo de programación de computadores. Se puede iniciar planteando a los estudiantes problemas matemáticos como los siguientes:

1. Luisa quiere invertir sus ahorros en la compra de discos compactos de moda. Si tiene $68.000, ¿Cuántos discos comprará?
Analizar el problema:
- ¿Qué tienes en cuenta cuando vas a comprar un disco?
- ¿Tienes información suficiente para resolver el problema de Luisa?
- ¿Qué dato averiguarías para saber cuántos discos puede comprar Luisa?

Plantear ahora este problema utilizando la metodología de "Formular el problema", "Resultados esperados", "Datos disponibles", "Determinar las restricciones" y "Procesos necesarios".

Cinco pasos que deben tener en cuenta los estudiantes para resolver problemas matemáticos:
1. Leer con mucho cuidado el problema hasta entenderlo.
2. Buscar la(s) pregunta(s).
3. Decidir lo que debes hacer.
4. Realizar las operaciones.
5. Comprobar que la respuesta hallada es correcta.

Pida a los estudiantes que contesten las siguientes preguntas en el proceso de solución de problemas matemáticos:

- ¿Cuántas preguntas tiene el problema? ¿Cuáles?
- ¿Qué debes hacer primero? ¿Para qué?
- ¿Qué debes hacer luego? ¿Para qué?
- ¿Cuál debe ser la respuesta (estimada) del problema?

Diseñar el algoritmo (trazar un plan)

Cuando se ha realizado un análisis a fondo del problema (utilizando alguna metodología), se puede proceder a elaborar el algoritmo (diagrama de flujo). Este consiste en la representación gráfica, mediante símbolos geométricos, de la secuencia lógica de las instrucciones (plan) que posteriormente serán traducidas a pseudocódigo y de ahí a un lenguaje de programación, como C, para ejecutarlas y probarlas en una computadora.

EJEMPLO
Diseñar un algoritmo (pseudocódigo y diagrama de flujo) para hallar el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.
R/
ANÁLISIS DEL PROBLEMA
Formular el problema: Ya se encuentra claramente planteado.
Resultados esperados: El área de un triángulo rectángulo.
Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo. La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir. El estudiante debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.
Determinar las restricciones: Utilizar las medidas dadas.
Procesos necesarios: Guardar en dos variables (BASE y ALTURA) los valores de Base y Altura; Guardar en una constante (DIV) el divisor 2; aplicar la fórmula BASE*ALTURA/DIV y guardar el resultado en la variable AREA; comunicar el resultado (AREA).

ALGORITMO EN PSEUDOCÓDIGO
Paso 1: Inicio
Paso 2: Asignar el número 2 a la constante "div"
Paso 3: Asignar el número 3 a la constante “base”
Paso 4: Asignar el número 4 a la constante “altura”
Paso 5: Guardar en la variable "área" el resultado de base*altura/div
Paso 6: Imprimir el valor de la variable "área"
Paso 7: Final

Traducir el algoritmo (ejecutar el plan)

Una vez que el algoritmo está diseñado y representado gráficamente se pasa a la etapa de traducción a un lenguaje de programación determinado (en nuestro caso será C). Cada lenguaje posee sus propias reglas gramaticales, por lo tanto es fundamental que los estudiantes conozcan de antemano la sintaxis de los comandos que deben utilizar para resolver el problema. A mayor dominio del lenguaje de programación, mayor posibilidad de llegar rápidamente a una solución satisfactoria. A esta fase de traducción se le conoce comúnmente como codificación.

Depurar el programa (revisar)

Después de traducir el algoritmo en un lenguaje de programación como C, el programa resultante debe ser probado y validados los resultados. A este proceso se le conoce como depuración. Depurar programas contribuye a mejorar la capacidad en los estudiantes para resolver problemas; la depuración basada en la retroalimentación es una habilidad útil para toda la vida.

Quienes han escrito alguna vez un programa de computadora, saben de la dificultad que representa elaborar programas perfectos en el primer intento, dificultad que aumenta a medida que el problema a resolver es más complejo. La depuración, afinamiento y documentación de un programa hacen parte fundamental del ciclo de programación y desde el punto de vista educativo estimula en los estudiantes la curiosidad, la perspectiva, la comunicación y promueve valores como responsabilidad, fortaleza, laboriosidad, paciencia y perseverancia. La programación facilita un diálogo interior en el cual la retroalimentación constante y el éxito gradual empujan a los alumnos a ir más allá de sus expectativas.

Otras dos actividades relacionadas con esta etapa son la afinación y la documentación. La primera consiste en realizar retoques para lograr una mejor apariencia del programa (en pantalla o en los resultados impresos) o para ofrecer funcionalidades más allá de los resultados esperados (especificados en la fase de análisis del problema). La segunda tiene un carácter eminentemente comunicativo, con la documentación de un programa se pone a prueba la capacidad del estudiante para informar a otras personas cómo funciona su programa y lo que significa cada elemento utilizado.

Conceptos básicos de programación

Variables

Para poder utilizar algoritmos con diferentes conjuntos de datos iniciales, se debe establecer una independencia clara entre los datos iniciales de un problema y la estructura de su solución. Esto se logra mediante la utilización de variables (cantidades que se suelen denotar con letras (identificadores) y que pueden tomar cualquier valor de un intervalo de valores posibles).
En programación, las variables son espacios de trabajo (contenedores) reservados para guardar datos (valores). El valor de una variable puede cambiar en algún paso del algoritmo o permanecer invariable; por lo tanto, el valor que contiene una variable es el del último dato asignado a ésta.
En C existen dos tipos de variables: locales y globales. Las primeras retienen su valor el tiempo que dure la ejecución del procedimiento en el cual se utilizan. Las variables locales crean dentro de una función o subrutina y sólo existen mientras se ejecuta la función.

Constantes

Las Constantes se crean en C por medio de una definición y consisten en datos que, luego de ser asignados, no cambian en ninguna instrucción del algoritmo. Pueden contener constantes matemáticas (pi) o generadas para guardar valores fijos (3.8, "Jorge", etc.).

Identificadores

Los identificadores son nombres que se dan a los elementos utilizados para resolver un problema y poder diferenciar unos de otros. Al asignar nombres (identificadores) a variables, constantes y procedimientos se deben tener en cuenta algunas reglas:

• Los nombres pueden estar formados por una combinación de letras y números (saldoMes, salario, fecha2, baseTriángulo, etc.).
• El primer carácter de un nombre debe ser una letra.
• La mayoría de los lenguajes de programación diferencian las mayúsculas de las minúsculas.
• Los nombres deben ser nemotécnicos, con solo leerlos se puede entender lo que contienen. Deben ser muy descriptivos; no utilizar abreviaturas, a menos que se justifique plenamente.
• Es conveniente utilizar una sola palabra para nombrar páginas, controles, variables, etc.
• No utilizar caracteres reservados (%, +, /, >, etc.).
• Se debe tener en cuenta que algunos lenguajes de programación no admiten las tildes.
• No utilizar palabras reservadas por los lenguajes de programación.
• Para cumplir con convenciones ampliamente utilizadas, los nombres de procedimientos, variables y constantes deben empezar con minúscula. Ejemplo, fecha, suma, etc.
• Si es un nombre compuesto por varias palabras, cada una de las palabras (con excepción de la primera) deben empezar con mayúscula. Ejemplo: fechaInicial, baseTriángulo, etc. (opcional).

Palabras reservadas (primitivas)

Todos los lenguajes de programación definen unas palabras para nombrar sus comandos, instrucciones y funciones. Un identificador definido por el usuario no puede tener el nombre de una palabra reservada en C.
Algunas palabras reservadas en C son:

ansi	continue	float	new	signed	try
auto	default for	operator	sizeof	typedef	
break	delete	friend	private	static	union
case	do	goto	protected	struct	unsigned
catch	double if	public	switch	virtual	
char	else	inline	register	template	void
class	enum	int	return	this	volatile
const	extern	long	short	throw	while

Subrutinas, procedimientos o funciones

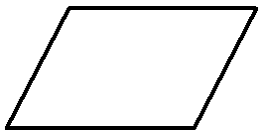
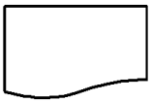
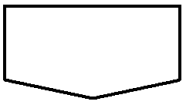
Una sub-rutina también conocidas como procedimientos o funciones son partes de un programa que se identifican por medio de un nombre y que realizan alguna actividad específica. Las subrutinas sirven para ahorrar código pues una actividad como por ejemplo sacar la raíz cuadrada de un número puede utilizarse en diferentes partes de un programa simplemente haciendo una llamada a una función en lugar de hacer todos los pasos necesarios para lograr obtener el resultado.

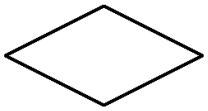
Las funciones reciben datos, los procesan y devuelven un resultado. Este resultado puede guardarse en una variable. Los datos que se le dan a una función para que trabaje con ellos se llaman parámetros. Cuando una función no regresa un resultado en algunos lenguajes se les llama procedimientos. En C todas las subrutinas se llaman funciones aunque no regresen un resultado.

Diagramas de flujo (Flujogramas):

Diagrama de flujo: Un diagrama de flujo o flujograma es una forma de representar gráficamente los pasos para resolver un problema en específico. Estos diagramas utilizan una serie de símbolos con significados especiales y son la representación gráfica de los pasos de un proceso. En computación es un primer enfoque con lo que sería la programación formal.

Simbología:

 INICIO/FIN	Este símbolo se utiliza para representar el inicio o el fin de un algoritmo. También puede representar una parada o una interrupción programada que sea necesaria al realizar un programa.
 PROCESO	Este símbolo se utiliza para un proceso determinado, es el que se utiliza comúnmente para representar una instrucción, o cualquier tipo de operación que origine un cambio de valor.
 ENTRADA/SALIDA	Este símbolo es utilizado para representar una entrada o salida de información, que sea procesada o registrada por medio de un periférico.
 IMPRESIÓN	Este símbolo se utiliza para mandar visualizar un resultado ya sea por medio de la pantalla o por una impresión.
	Este símbolo es utilizado para enlazar dos partes del diagrama pero en diferentes hojas.

CONECTOR FUERA DE PAGINA	
 DECISION	Este es utilizado para la toma de decisiones, ramificaciones, para la indicación de operaciones lógicas o de comparación entre datos.
 CONECTOR	Este símbolo es utilizado para enlazar dos partes cualesquiera de un diagrama a través de un conector de salida y un conector de entrada. Esta forma un enlace en la misma página del diagrama.

Aspectos a tomar en cuenta para crear un buen diagrama de flujo:

- Existe siempre un camino que permite llegar a una solución (finalización del algoritmo).
- Existe un único inicio del proceso.
- Existe un único punto de fin para el proceso de flujo (salvo del rombo que indica una comparación con dos caminos posibles).

Desarrollo del Diagrama de Flujo:

Acciones previas a la realización del diagrama de flujo:

- Definir qué se espera obtener del diagrama de flujo.
- Identificar quién lo empleará y cómo.
- Establecer el nivel de detalle requerido.
- Determinar los límites del proceso a describir.

Pasos a seguir para construir el diagrama de flujo:

- Establecer el alcance del proceso a describir. De esta manera quedará fijado el comienzo y el final del diagrama. Frecuentemente el comienzo es la salida del proceso previo y el final la entrada al proceso siguiente.
- Identificar y listar las principales actividades/subprocesos que están incluidos en el proceso a describir y su orden cronológico.
- Si el nivel de detalle definido incluye actividades menores, listarlas también.
- Identificar y listar los puntos de decisión.

- Construir el diagrama respetando la secuencia cronológica y asignando los correspondientes símbolos.
- Asignar un título al diagrama y verificar que esté completo y describa con exactitud el proceso elegido.

Ventajas de Diagramas de flujo:

- Favorecen la comprensión del proceso a través de mostrarlo como un dibujo. El cerebro humano reconoce fácilmente los dibujos. Un buen diagrama de flujo reemplaza varias páginas de texto.
- Permiten identificar los problemas y las oportunidades de mejora del proceso. Se identifican los pasos redundantes, los flujos de los re-procesos, los conflictos de autoridad, las responsabilidades, los cuellos de botella, y los puntos de decisión.
- También dentro del ámbito laboral más allá de la ingeniería muestran la interacción de cliente-proveedor así como la información que en ellas se realizan, facilitando a los empleados el análisis de las mismas para futuros procesos o capacitaciones más rápidas dentro de un entorno general de trabajo.
- Son una excelente herramienta para capacitar a los nuevos empleados y también a los que desarrollan la tarea, cuando se realizan mejoras en el proceso.

 El diagrama de flujo representa lo que es una secuencia de pasos a seguir por medio de símbolos y seguidos en secuencia por medio de conectores en flechas llegando a una solución para una entrada; es una relación finita dado que siempre llegara a un fin.

Pseudocódigo:

Pseudocódigo: Un pseudocódigo (falso lenguaje), es una serie de instrucciones a seguir pero utilizando palabras léxicas y gramaticales referidos a los lenguajes de programación, pero sin llegar estar estrictamente correcta su sintaxis de programación; ni tener la fluidez del lenguaje coloquial. Permitiendo codificar un programa con mayor agilidad que en cualquier lenguaje de programación. Forma parte de las distintas herramientas de la ingeniería de software y es, netamente, lenguaje de tipo informático.

La estructura de un pseudocódigo puede ser de tres tipos:

- Secuencial.- esta consiste en colocar cada instrucción una tras de la otra sin tener ningún tipo de saltos
- Selectiva.- esta lleva a cabo ciertas instrucciones cuando se cumple una cierta condición y si esta condición no se cumple se salta a la siguiente instrucción.
    - Selectiva doble (anidamiento).- Esta realiza una instrucción u otra según la respuesta de la condición planteada
    - Selectiva múltiple.- Esta realiza instrucciones para distintos comportamientos de las condiciones, que sería como trabajar varias selectivas dobles.
    - Selectiva múltiple-casos.- Esta realizara para un cierto tipo de or declarado en un inicio y dependiendo cual sea será el tipo de comportamiento a realizar
- Iterativa.- Este consiste en la posibilidad de realizar una misma instrucción más de una vez
    - Bucle mientras.- Realiza ciertas instrucciones mientras que la condición se siga cumpliendo
    - Bucle repetir.- Realiza ciertas instrucciones hasta que se deje de cumplir con la condición que a diferencia del mientras esta instrucción realiza al menos una vez las instrucciones
    - Bucle para.- Se utiliza para realizar instrucciones cierto número de veces pero definiendo por un índice que se incrementa en cada vuelta
    - Bucle para cada.- Realiza instrucciones para todo elemento que cumpla con la condición.

 Al igual que en el caso del diagrama de flujo existen ciertas reglas para que sea un buen pseudocódigo:

1.-Tenga un único punto de inicio

2.-Tenga un número finito de posibles puntos de término

3.-Haya un número finito de caminos, entre el punto de inicio y los posibles puntos de termino

Funciones y Procedimientos:

Al plantear y resolver algoritmos medio de la programación, al estar trabajándolos saltan a la vista dos conceptos que aunque son similares no son lo mismo, los cuales son las funciones y los procedimientos.

Una función al igual que en matemáticas recibe uno o más posibles valores y entrega una salida, y un procedimiento recibe una entrada pero no genera salida.

Un procedimiento son instrucciones con las cuales realizaras un proceso pero son como una receta y no ocupas nada de símbolos matemáticos.

Características de Pseudocódigos

- Ocupan mucho menos espacio en el desarrollo del problema.
- Permite representar de forma fácil operaciones repetitivas complejas.
- Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.
- En los procesos de aprendizaje de los alumnos de programación, estos están más cerca del paso siguiente (codificación en un lenguaje determinado, que los que se inician en esto con la modalidad Diagramas de Flujo).
- Mejora la claridad de la solución de un problema

 Todo documento en pseudocódigo debe permitir la descripción de:

- Instrucciones primitivas: Asignación (5→A), Entrada de datos, Salida de datos.
- Instrucciones de proceso: Ciclos, Operaciones con datos
- Instrucciones compuestas: Desplegar algo en pantalla, ocupar una dirección como referencia y trabajar con ella
- Instrucciones de descripción: mensajes para usuario de que seguiría dentro del programa

Estructura a seguir en su realización

Cabecera:

- Programa:
- Módulos:

- Tipos de datos:
- Constantes:
- Variables:

Cuerpo:

- Inicio
- Instrucciones
- Fin

 Para comentar en pseudocódigo se le antepone al comentario dos asteriscos (**)

Ejemplos:

** Programa que calcula el área de un cuadrado a partir de un lado dado por teclado.*

** Programa que visualice la tabla de multiplicar del número introducido por teclado*

El pseudocódigo es otra forma de representar una solución a un problema con la diferencia que se hace ya un poco más parecido a lo que sería ya un lenguaje de programación pero ser estrictos en la sintaxis, tiene distintas formas de estructurarse y sus distintas secuencias, iteraciones y repeticiones son ya muy similares al menos en contexto a lo que serian ya dentro de un programa.

Sintaxis del Pseudocódigo

Si <condcion>

Entonces

Fin_Si

Si <condición>

Entonces

Instrucciones

```
        Si no
                Instrucciones
Fin_Si
```

```
Si <condición>
        Entonces
                Instrucciones
        Si no
                Si <condicon>
                Entonces
                        Instrucciones
                Si no
                        Instrucciones
                Fin_Si
Fin_Si
```

```
En caso de <expresión>
        Haga
                Caso <opción 1>:
                        Instrucciones
                Casi <opción 2>:
                        Instrucciones
        Si no
                Instrucciones
Fin_caso
```

```
Mientras que <condición>
        Instrucciones
Fin_Mientras
```

```
Repita
        Instrucciones
```

Hasta <condición>

Para <var> = <exp1> hasta <exp2> paso <exp3> haga

    Instrucciones

Fin_para

Prueba de escritorio:

Prueba de escritorio: Una prueba de escritorio es la comprobación lógica, de un algoritmo de resolución.

Para desarrollar la prueba de escritorio, se utilizara el siguiente procedimiento:

- Con datos de prueba, se seguirán cada uno de los pasos propuestos en el algoritmo de resolución.
- Si la prueba de escritorio genera resultados óptimos, quiere decir que el algoritmo posee una lógica adecuada, en caso contrario el algoritmo tendrá que ser corregido.

Ejemplo:

```
suma :entero
entrada :entero
menor :entero

leer entrada
menor = entrada
suma = 0
mientras (entrada != 0) haga
si (entrada < menor) entonces
menor =entrada
fin_si
suma = suma + entrada
leer entrada
fin_mientras
escribir "valor Menor:"
escribir menor
```

escribir “Suma:”
escribir suma

INSTRUCCIÓN	entrada	menor	suma	Pantalla
leer entrada				
menor = entrada				
suma :=0				
suma :=suma + entrada				
leer entrada				
menor = entrada				
suma :=suma + entrada				
leer entrada				
suma :=suma + entrada				
leer entrada				
Escribir “valor menor:”				
Escribir menor				
Escribir “Suma:”				
Escribir suma				

Tareas por asignar o Ejercicios

Inserte aquí las tareas o ejercicios a realizar por los estudiantes

- Tarea I
  Realizar un pseudocódigo para los siguientes problemas tomar en cuenta todos los posibles casos a ocurrir:
  1.-Calcular el año en que nació una persona a partir de su año y mes de nacimiento
- Tarea II
  Realizar las respectivas pruebas de escritorio para cada uno de los ejercicios y en cada caso completar lo que falte (a diagramas su pseudocódigo y viceversa)

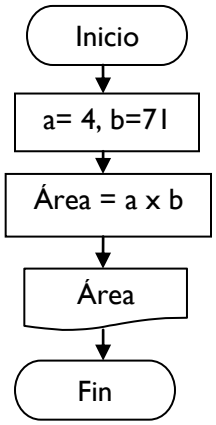
Ejemplos en clase

Calcular el área de un rectángulo de lados con valor de 4 x 71

a=4 , b=71

int Área

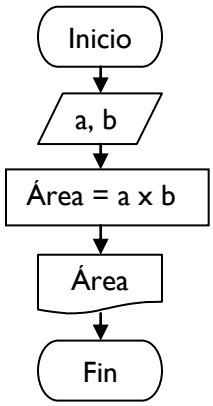
1. Inicio
2. Área= a x b
3. Mostrar "Área"
4. Fin



Calcular el área de un rectángulo de cualquier tamaño

int Área, a, b

1. Inicio
2. Área= a x b
3. Mostrar "Área"
4. Fin



Cálculo de grados centígrados a Fahrenheit o viceversa

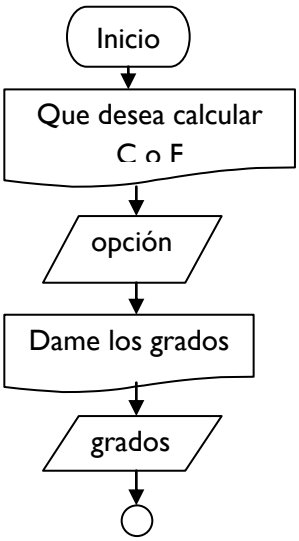
Float resul, grados

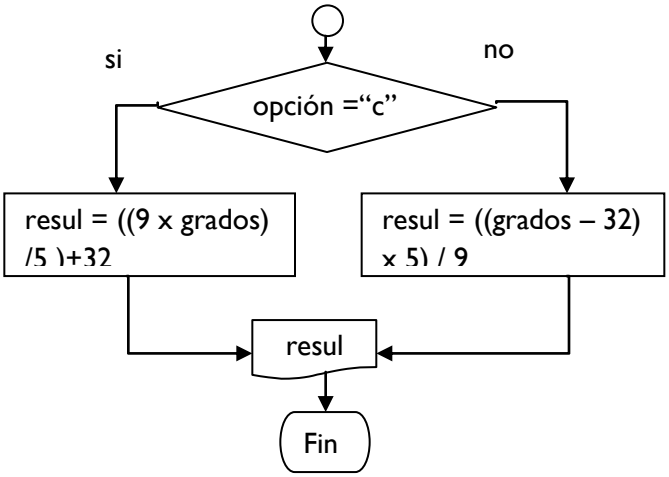
Char opción

1. Inicio
2. Mostrar " Ingrese que desea calcular C o F"
3. Leer opción
4. Mostrar "Dame los grados"
5. Leer grados
6. Si (opción="C")
   Entonces
   resul = ((9 x grados) /5)+32
   Si no
   resul = ((grados – 32) x 5) / 9

   Fin_Si

5. Mostrar resul
6. Fin

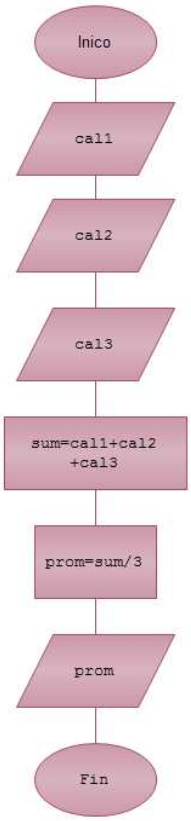




Obtener el valor del promedio de 3 calificaciones

Float a, b, c, sum, prom

1. Inicio
2. Mostrar dame las 3 calificaciones
3. Leer a, b, c
4. Sum = a + b + c
5. Prom = Sum / 3
6. Mostrar "el promedio es", Prom
7. Fin



Obtener el resultado de la resta de un número menos un número más grande que el otro

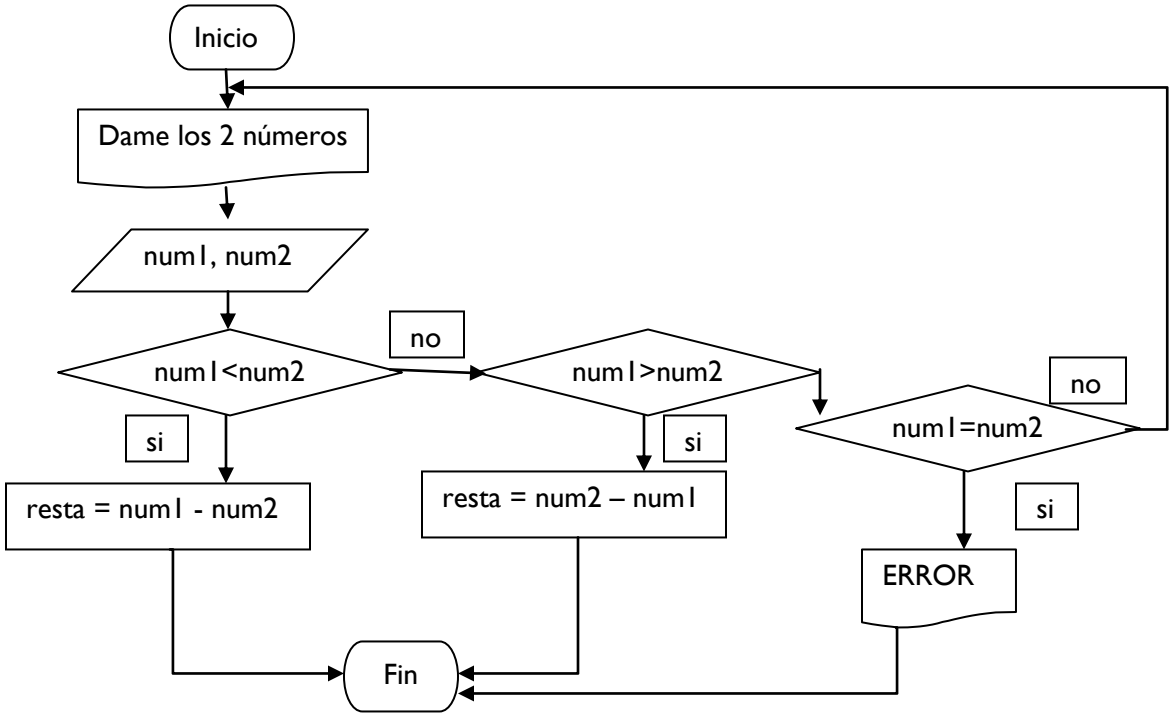
int num1, num2, resta

1. Inicio
2. Mostrar "Dame los 2 números"
3. Leer num1 y num2
4. Si (num1<num2)
   Entonces
   resta = num1 - num2
   Si no
   Si (num1>num2)
   entonces
   resta =num2 – num1
   Si no
   Si (num1=num2)
   entonces
   Mostrar "0"
   Si no
   Mostrar "error"
   Ir a paso 2

Fin_Si

Fin_Si

Fin_Si



Que el usuario de su género y su nombre, dependiendo de su género imprimir en pantalla Bienvenido "nombre", Bienvenida "nombre", y si en dudo caso se pone otra opción imprimir en pantalla Hola indeciso.

Char sexo, nombre

5. Inicio
6. Leer sexo
7. Leer nombre
8. Si (sexo = hombre)

Entonces

Mostrar "Bienvenido", nombre

si no

Si (sexo == mujer)

Entonces

Mostrar "Bienvenida", nombre

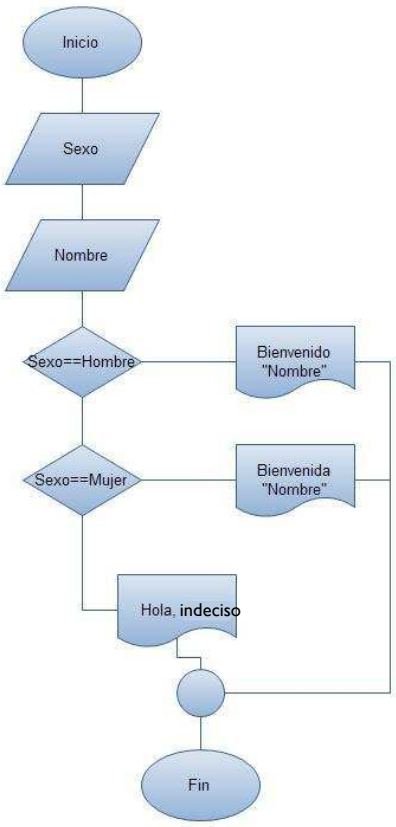
si no

Mostrar “Hola indeciso”

Fin_Si

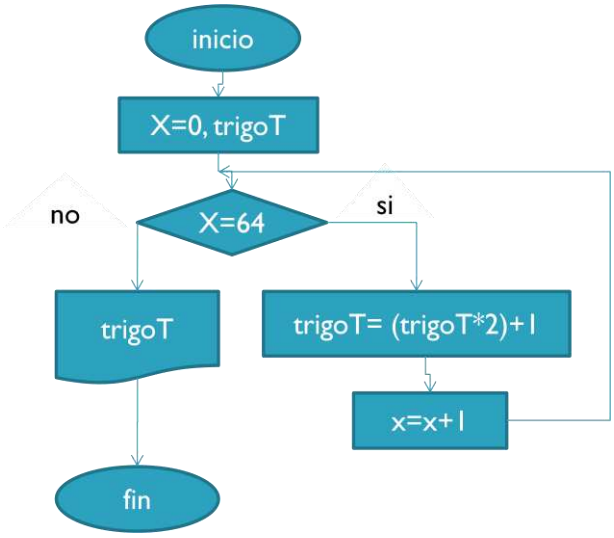
Fin_Si

9. Fin



Problema del ajedrez y el trigo
Una persona le pidió al rey que le pusiese en cada casilla de ajedrez granos de arroz, con la lógica que en cada casilla tuviese el doble de granos que la anterior

```
Int : x, trigoT
1-Inicio
2-trigoT=0
3-Para x=0 hasta x=63 paso 1
        haga
                Calcula trigoT= (trigoT*2) + 1
Fin_Para
5- Imprime trigoT
4-Fin
```

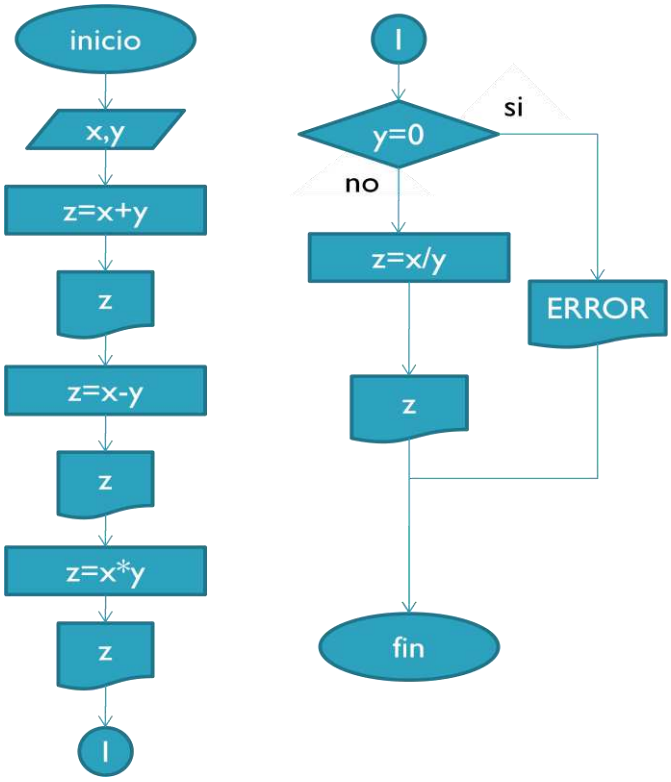


Cálculo de operaciones con solo 3 variables y obtener la suma, la resta, la división y la multiplicación

```
Int x, y , z
1. Inicio
2. Mostrar "Dame 2 datos"
3. Leer x, y
4. Calcular z= x + y
5. Mostrar z
6. Calcular z= x - y
7. Mostrar z
8. Calcular z= x * y
9. Mostrar z
10. Si y=0
                Entonces
                        Mostrar Error
                Si no
                        z= x / y
                        Mostrar z
```

Fin_Si

11. Fin

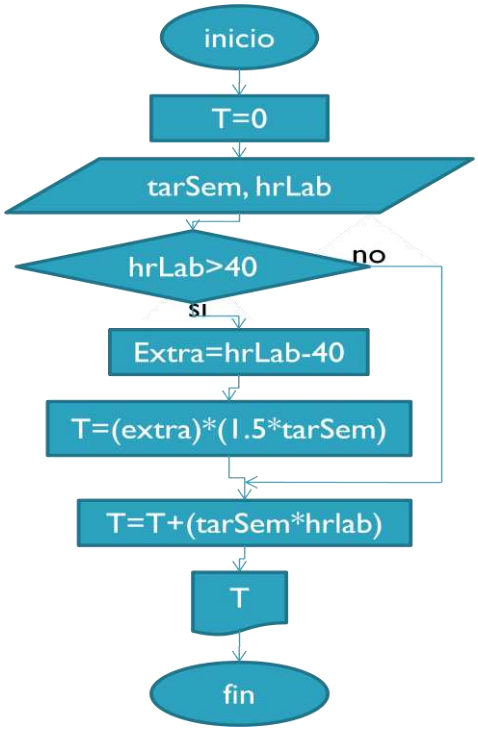


Pago de un empleado según sus horas de trabajo

Int hr_lab, tar_sem, T, extra

1. Inicio
2. T=0
3. Mostrar "Dame el valor de la tarifa semanal"
4. Leer tarSem
5. Mostrar "Dame el valor de horas laboradas"
6. Leer hrLab
7. Si (hrLab>40)
   Entonces
   Calcular extra=hr_lab-40
   Calcular T= extra*(1.5*tarSem)

   Fin_Si
8. T=T+(tarSem*hrLab)
9. Mostrar T
10. Fin



Adivina el numero que está pensando la computadora

```
a=6,  int num
    1. Inicio
    2. Repita
                Mostrar "Dame un numero"
                Leer num
                Si a != num
                        Entonces
                                Si a>num
                                        Entonces
                                                Mostrar "Mi número es mayor"
                                        Si no
                                                Si a<num
                                                        entonces
                                                                Mostrar "Mi número es menor"
                                                        Si no
                                                        Mostrar "lo que diste no es un numero"
                                                Fin_Si
                                Fin_Si
                Fin_Si
        Hasta (a!=num)

    3. Mostrar  "Felicidades, ya ganaste"
    4. Fin
```

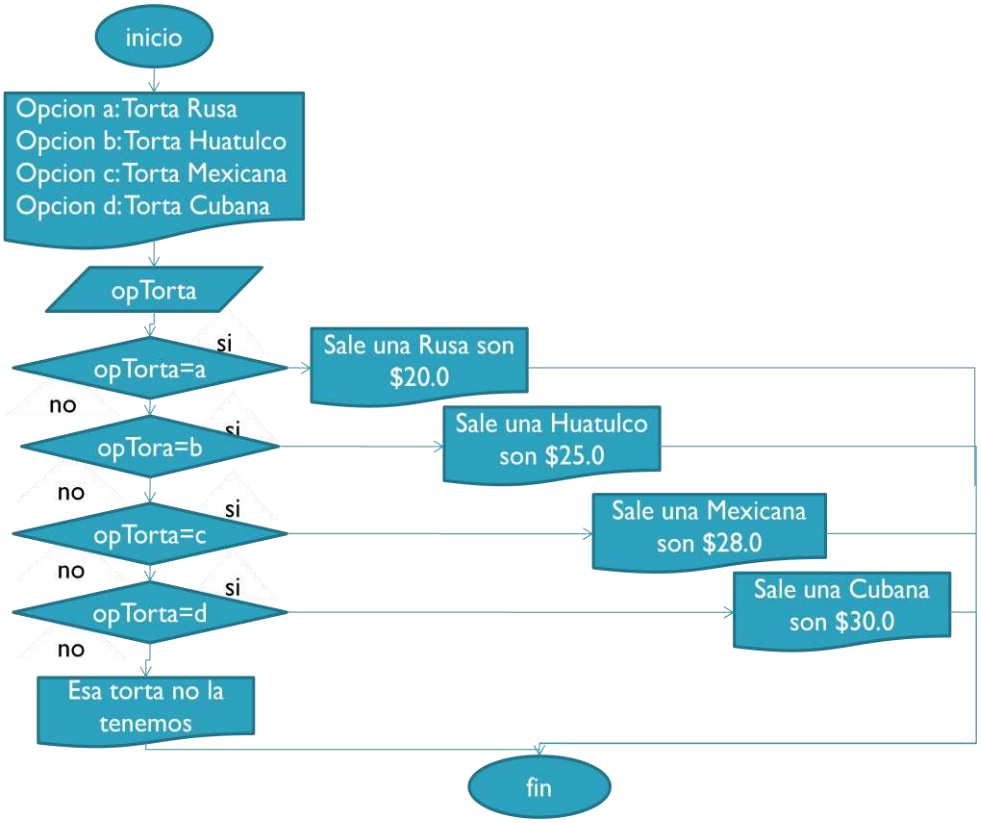
Un menú que te permite decidir que torta quieres

Char OpTorta

1. Inicio
2. Mostrar "Opcion a: Torta rusa"
   "Opcion b: Torta Huatulco"
   "Opcion c: Torta Mexicana"
   "Opcion d: Torta Cubana"
3. Leer OpTorta
4. En caso de (OpTorta) haga
   Caso <a>:
   Mostrar " Sale una torta Rusa!! Son $20
   Caso <b>:
   Mostrar " Sale una torta Huatulco!! Son $25
   Caso <c>:
   Mostrar "Sale una torta Mexicana!! Son $28
   Caso <d>:
   Mostrar "Sale una torta Cubana!! Son $30
   Si no
   Mostrar " Esa opción no la tenemos"

   Fin_Caso

5. Fin



Una empresa necesita tener un reloj de 24 horas, solo se quiere que muestre, horas, minutos y segundos

Hacer un reloj de 24 horas

int hr, min, seg

1. Inicio
2. Para hr=0 hasta hr=23 paso 1 haga

Para min=0 hasta min 59 paso 1 haga

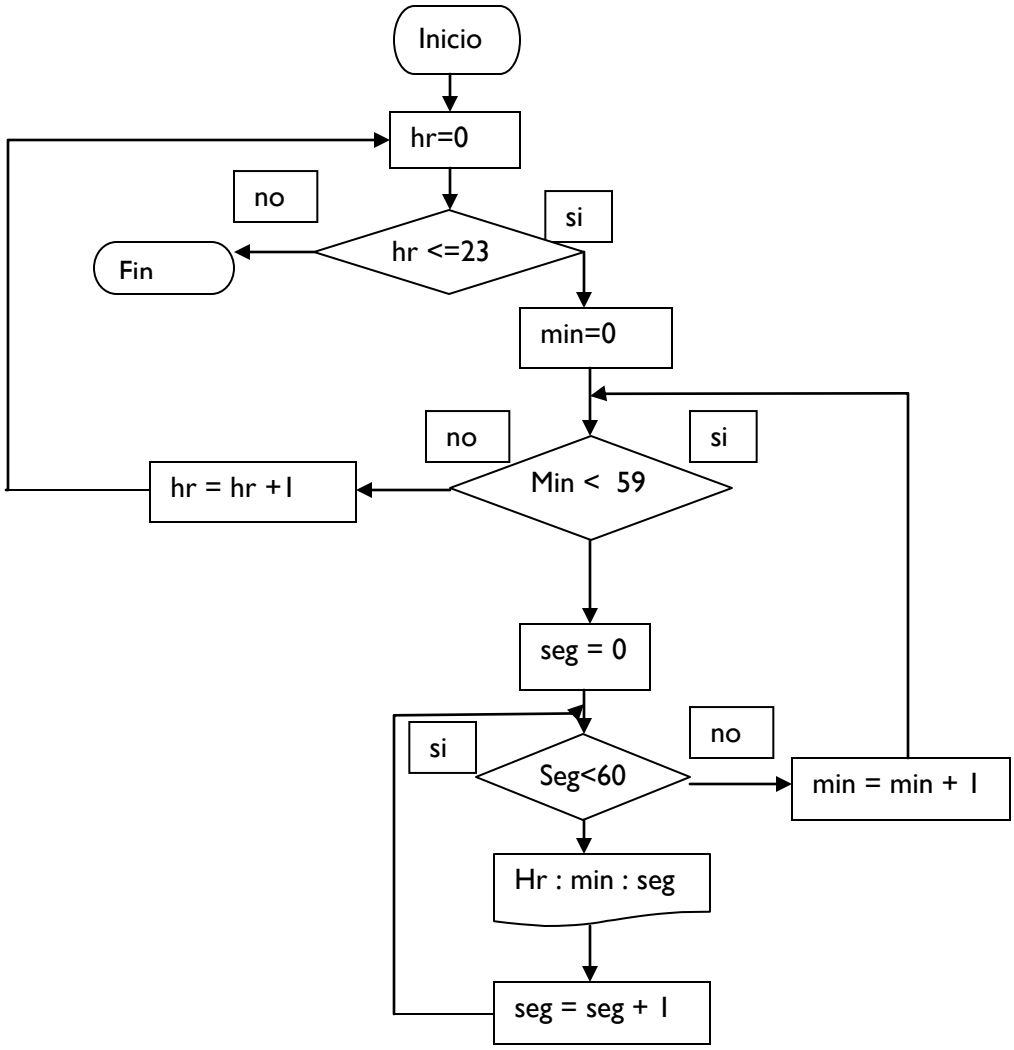
Para seg00 hasta seg=59 paso 1 haga

Mostrar hr: min: seg

Fin_Para

Fin_Para

3. Fin



Obtener el factorial de un numero

1. Inicio
2. Mostar “dame el numeo”
3. Leer N
4. F=1
5. Si (N>1)

entonces

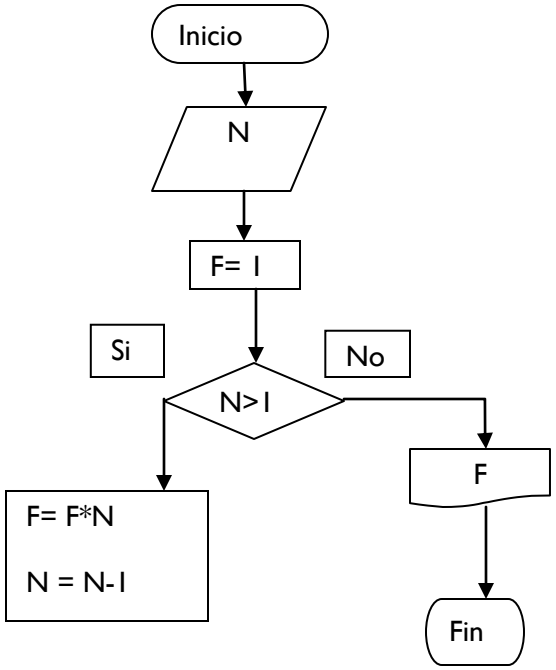
Calcular F=F*N

Calcular N=N-1

si no

Mostrar F

6. FIN



Int existen, sobran=60, compras= 0, boleto, aux=0

Algoritmo que requiere una cadena de cines para vender 60 entradas a una sala con pantalla de 360°

1. Inicio
2. Sobran=60, compras =0, aux=0
3. Mientras sobran> 0

    Mientras compras <61 entonces

    Muestra “cuantos boletos quiere??”

    Leer boleto

existen= sobran

calcula sobran= 60- compras

si sobran> 0 entonces

compras = compras + boletos

sobran= sobran– compras

si compras>60 entonces

aux= 60- compras

sobran=aux

fin_si/*cuando quieren comprar mas de los disponibles*/

fin_si

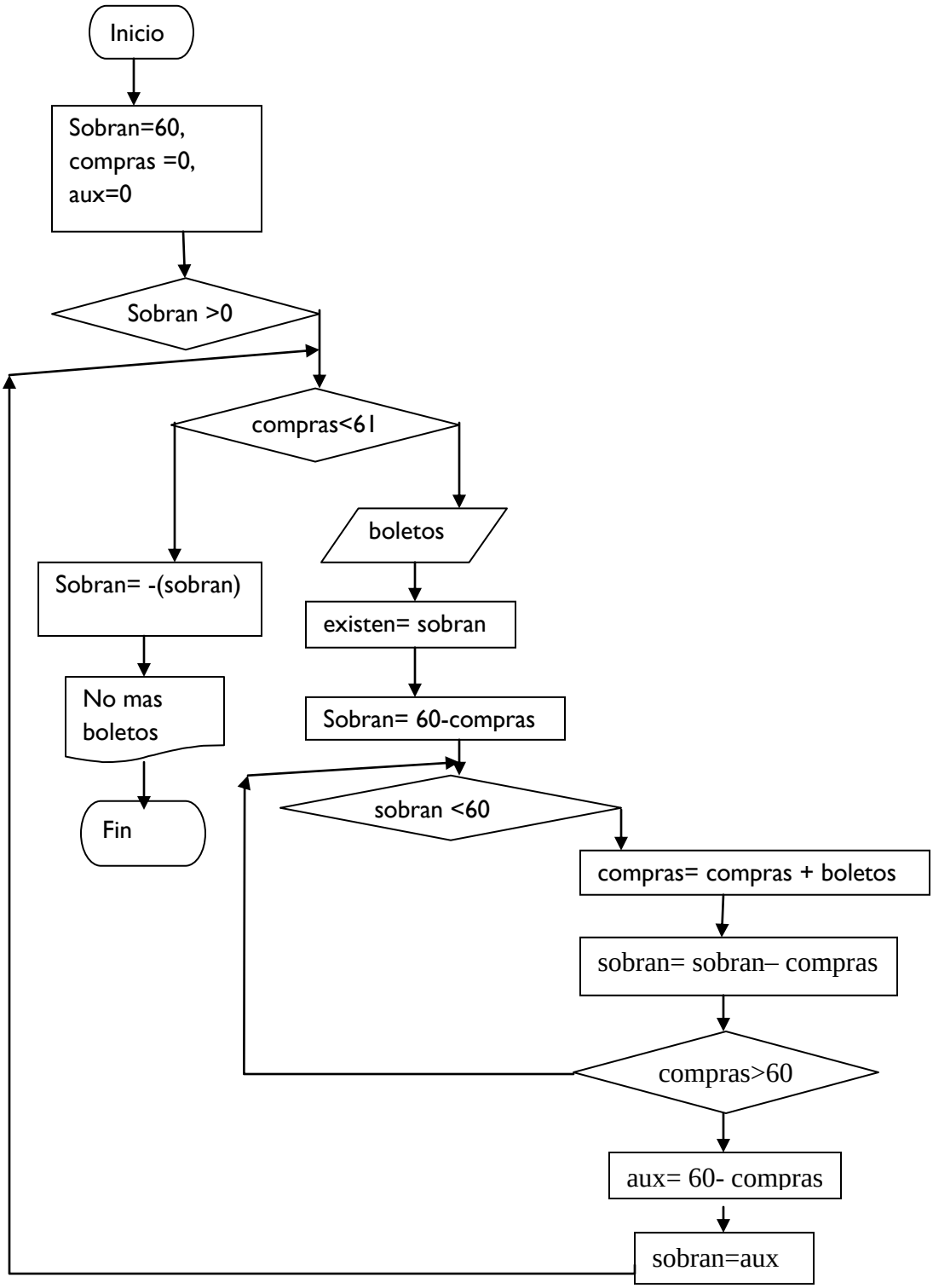
fin_mientras

fin_mientras

sobran= -sobran

4. Muestra “lo sentimos se agotaron las entradas solo nos falto venderte “sobran” boletos”

Fin


Inicio
Sobran=60,
compras =0,
aux=0
Sobran >0
compras<61
boletos
Sobran= -(sobran)
existen= sobran
No mas
boletos
Sobran= 60-compras
Fin
sobran <60
compras= compras + boletos
sobran= sobran– compras
compras>60
aux= 60- compras
sobran=aux

Material derivado de:

ALGORITMOS Y PROGRAMACIÓN (GUÍA PARA DOCENTES)
SEGUNDA EDICIÓN, 2007, 2009.
Juan Carlos López García
http://www.eduteka.org

Control de Versión

Autor	Fecha	Versión
Ian Coronado	10-ix-2009	1.1
Priscilla Hdz	22-ix-2009	Revisión
Ian Coronado	23-ix-2009	2.0
Ian Coronado	06-i-2010	3.0
Honorato Saavedra	12-1-2020	3.1