- **¿Qué es una computadora?**

Es una máquina electrónica capaz de recibir, procesar y almacenar datos para producir información útil mediante programas.

- **¿Dónde debe cargarse un programa para que se pueda ejecutar?**

En la **memoria principal** (RAM), ya que el procesador solo puede ejecutar instrucciones que estén allí.

- **Enumere y explique los dos conceptos asociados a una celda de memoria**

  - **Dirección**: Identificador único de la celda, como su "nombre".
  - **Contenido**: Dato almacenado en esa celda.

- **¿Qué se guarda en la memoria principal?**

Se almacenan temporalmente programas en ejecución, datos utilizados por esos programas y el sistema operativo.

- **¿Qué es un algoritmo?**

Es una secuencia ordenada y finita de pasos que resuelven un problema o realizan una tarea específica.

- **¿Cuáles son los pasos para la resolución de un problema?**

  1. Comprender el problema
  2. Analizar los datos
  3. Diseñar un algoritmo
  4. Codificar el algoritmo
  5. Probar y depurar
  6. Documentar y mantener

- **Enumere las características de un algoritmo**

  - **Finito**: Tiene un número limitado de pasos
  - **Definido**: Cada paso debe ser claro y no ambiguo
  - **Ordenado**: Pasos en secuencia lógica
  - **Efectivo**: Cada paso debe ser realizable
  - **Entrada y salida**: Tiene datos de entrada y genera resultados

- **¿Cuáles son los tipos de lenguajes?**

  - **Lenguaje de máquina**
  - **Lenguaje de bajo nivel (ensamblador)**
  - **Lenguaje de alto nivel (C, Python, Java, etc.)**

- **¿Cuáles son los tipos de instrucciones?**

  - **Instrucciones de entrada/salida**
  - **Instrucciones de procesamiento (cálculo)**
  - **Instrucciones de control (saltos, bucles)**

- **Instrucciones de almacenamiento (asignaciones)**

- **¿Qué es un lenguaje de máquina?**

Es el conjunto de instrucciones que entiende directamente la CPU. Está compuesto por números binarios (0 y 1).

- **¿Qué es un lenguaje de bajo nivel?**

Es un lenguaje cercano al lenguaje de máquina, como el ensamblador. Requiere conocimiento del hardware.

- **¿Qué es un lenguaje de alto nivel?**

Es un lenguaje de programación más cercano al lenguaje humano, más fácil de escribir, leer y mantener. Ejemplos: C, Python, Java.

- **¿Qué diferencia hay entre un intérprete y un compilador?**

  - **Intérprete**: Traduce y ejecuta el código línea por línea.
  - **Compilador**: Traduce todo el código a lenguaje máquina antes de ejecutarlo.

- **Enumere las fases de la compilación**

  1. Análisis léxico
  2. Análisis sintáctico
  3. Análisis semántico
  4. Generación de código intermedio
  5. Optimización de código
  6. Generación de código objeto
  7. Enlazado (linking)

- **Explique brevemente la historia del lenguaje C y sus ventajas**

C fue desarrollado en los años 70 por Dennis Ritchie en los laboratorios Bell para el sistema operativo UNIX.
Ventajas:

  - Portabilidad
  - Eficiencia y control de hardware
  - Sintaxis clara
  - Base para otros lenguajes (como C++, Java)
  - Amplio uso en sistemas operativos, embebidos y software de alto rendimiento

1. **Enumere y explique brevemente las fases en la resolución de problemas**
   - **Análisis del problema**: Comprender el problema, sus requisitos y restricciones.
   - **Diseño del algoritmo**: Crear una solución paso a paso.
   - **Codificación**: Traducir el algoritmo a un lenguaje de programación.
   - **Prueba y depuración**: Ejecutar el programa, corregir errores.
   - **Documentación y mantenimiento**: Explicar cómo funciona el programa y actualizarlo si es necesario.

2. **¿Qué es un algoritmo?**
   Es una secuencia de pasos ordenados, finitos y lógicos que resuelven un problema o realizan una tarea.
3. **¿Cuáles son las preguntas para definir un problema en la etapa del análisis?**
   - ¿Qué datos se conocen?
   - ¿Qué se debe obtener como resultado?
   - ¿Qué operaciones o procesos se deben realizar?
   - ¿Qué restricciones existen?
4. **¿Cuál es el diseño descendente?**
   Es una técnica que consiste en dividir un problema grande en subproblemas más pequeños y manejables, de forma jerárquica.
5. **Explique diagramas de flujo y pseudocódigo**
   - **Diagrama de flujo**: Representación gráfica del algoritmo con símbolos que muestran el flujo del proceso.
   - **Pseudocódigo**: Representación escrita del algoritmo en lenguaje natural estructurado, similar a un lenguaje de programación.
6. **Explique la documentación interna y la externa de un programa**
   - **Interna**: Comentarios en el código fuente que explican su funcionamiento.
   - **Externa**: Manuales o documentos que explican cómo usar el programa o cómo está diseñado.
7. **¿Qué tipos de errores hay?**
   - **Sintácticos**: Fallas en la escritura del código (reglas del lenguaje).
   - **Semánticos**: El código está bien escrito pero no hace lo que se espera.
   - **Lógicos**: El algoritmo es incorrecto y da resultados equivocados.
   - **De ejecución**: Errores que ocurren al correr el programa (división por cero, archivos no encontrados, etc.).
8. **¿Qué es la programación modular?**
   Es una técnica que divide un programa en módulos o funciones independientes que se pueden desarrollar y probar por separado.
9. **¿Qué es la programación estructurada?**
   Es un paradigma que organiza el código usando estructuras de control (secuencia, selección, repetición) para hacerlo más legible y mantenible.
10. **¿Qué técnicas incorpora la programación estructurada?**

- **Secuencia**: Instrucciones ejecutadas en orden.
- **Selección**: Decisiones mediante condicionales (if, else).
- **Repetición**: Bucles para repetir instrucciones (for, while).
- **Modularidad**: División en funciones o módulos.

1. **¿Qué dice el teorema de la programación estructurada?**
   Establece que **cualquier algoritmo** puede construirse usando solo **tres estructuras básicas**: **secuencia, selección e iteración**.
2. **¿Qué función no falta nunca en un programa escrito en C?**
   La función `main()` —es el punto de inicio donde comienza la ejecución del programa.

3. **¿Qué es una función?**
   Es un bloque de código que realiza una tarea específica y puede ser llamado desde otras partes del programa.
4. **¿Qué es una función definida por el usuario?**
   Es una función creada por el programador para realizar tareas personalizadas, distintas de las funciones estándar del lenguaje.
5. **Tipos de datos más comunes en C y sus máscaras de entrada/salida:**

Tipo	Descripción	Máscara entrada/salida
`int`	Enteros	`%d`
`float`	Reales simples	`%f`
`double`	Reales dobles	`%lf`
`char`	Caracteres	`%c`
`char[]`	Cadenas de texto	`%s`

6. **Rango que cubren distintos tipos de datos:** (puede variar por sistema)
   - `char`: -128 a 127
   - `unsigned char`: 0 a 255
   - `int`: -32,768 a 32,767 (en sistemas de 16 bits)
   - `unsigned int`: 0 a 65,535
   - `float`: ±3.4 × 10^38 (6 decimales)
   - `double`: ±1.7 × 10^308 (15 decimales)
7. **¿Cómo deben nombrarse las variables en C?**
   - Deben comenzar con una letra o guion bajo (_)
   - Solo pueden contener letras, números y guiones bajos
   - No pueden ser palabras reservadas
   - Deben ser descriptivos y sin espacios
8. **¿Cuáles son los operadores especiales aritméticos?**
   - `++`: incremento
   - `--`: decremento
   - `%`: módulo (resto de división entera)
9. **Tipos de estructuras en lenguaje C:**
   - **Secuenciales**
   - **Selectivas** (condicionales)
   - **Repetitivas** (bucles)
   - **Compuestas** (como estructuras `struct`)
10. **Sentencias que se dan en cada una:**

- **Secuencial**: ejecución de instrucciones una tras otra
- **Selectiva**: `if`, `else`, `switch`
- **Repetitiva**: `for`, `while`, `do...while`

11. **Tipos de funciones:**

- Con retorno y con parámetros
- Con retorno y sin parámetros
- Sin retorno y con parámetros
- Sin retorno y sin parámetros

12. **Partes a declarar de una función:**

- **Tipo de retorno**
- **Nombre de la función**
- **Parámetros** (si hay)
- **Cuerpo de la función** (bloque de instrucciones)

13. **Ámbito de una variable:**

- **Local**: declarada dentro de una función o bloque
- **Global**: declarada fuera de cualquier función
- **Estática**: mantiene su valor entre llamadas si se usa `static`

14. **¿En qué difiere si declaro una variable antes del main, dentro del main o dentro de una función creada por el usuario?**

- **Antes del `main`**: es **global**, accesible desde cualquier función
- **Dentro del `main`**: es **local al `main`**
- **Dentro de otra función**: es **local a esa función**

15. **¿La variable descripta en el main es la misma que la descripta dentro de una función?**
    No, son **variables distintas** con **ámbitos diferentes**. Aunque tengan el mismo nombre, no comparten valor.
16. **Indique los pasos a desarrollar para escribir y hacer funcionar un programa en C:**
17. Escribir el código fuente en un editor
18. Guardar con extensión `.c`
19. Compilar el código con un compilador (ej. `gcc`)
20. Corregir errores de compilación
21. Ejecutar el archivo generado
22. **Indique los errores más comunes en C:**

- Olvidar punto y coma `;`
- Uso incorrecto de tipos de datos
- No declarar variables
- Mal uso de operadores lógicos o aritméticos
- Acceder a índices fuera de un arreglo
- Errores de compilación por mala sintaxis
- Errores de ejecución como división por cero