1. Definir el concepto de programa de computadora.
2. Qué entiende por algoritmo.
3. Que es un lenguaje de programación de alto nivel.
4. Describir los procesos de compilación y ejecución.
5. Brevemente Describir la historia del lenguaje C++.
6. Para los recién iniciados, describir los principales componentes de una computadora y cómo funcionan juntos.
7. Por qué debemos aprender acerca de algunos asuntos éticos básicos que enfrentan los profesionales de la computación.
8. Podrías Listar las etapas básicas relacionadas con la escritura de un programa de computadora.
9. Describir lo que es un compilador y lo que hace.
10. Para los recién iniciados, diferenciar entre hardware y software.
11. Describir un método apropiado de resolución de problemas para desarrollar una solución algorítmica a un problema.
12. Programa de computadora: Secuencia de instrucciones que realizará una computadora.
13. Algoritmo: secuencia ordenada de pasos que sirven para realizar una tarea en una cantidad finita de tiempo.
14. Lenguaje de programación Conjunto de reglas, símbolos y palabras especiales usado para implementar un programa de computadora. Debido al funcionamiento binario de los componentes de la computadora, en lenguaje de máquina todo se representa por ceros y unos. Dado que esto es sumamente tedioso y puede conducir a errores, se diseñó primero un lenguaje ensamblador para una comunicación más sencilla. Luego aparecieron los lenguajes de alto nivel. Estos son más fáciles de usar, se asemejan más al lenguaje natural. Permiten codificar los algoritmos de una forma más cercana a la capacidad cognitiva humana.
15. Compilación: Traducción del lenguaje de alto nivel al código de máquina. Durante la ejecución, el programa objeto se carga en la memoria de la computadora. De esta manera, la computadora ejecuta el programa objeto y las instrucciones del programa.
16. A finales de la década de 1960 y principios de la de 1970, Dennis Ritchie creó el lenguaje de programación C en AT&T Bell Labs. En aquella época, un grupo de personas en los laboratorios Bell estaba diseñando el sistema operativo UNIX. Al inicio, UNIX se escribía en lenguaje ensamblador, como era costumbre para casi todo el software de sistema en esos días. Para evitar las dificultades de programar en lenguaje ensamblador, Ritchie inventó C como un lenguaje de programación de sistema. C combina las características de bajo nivel de un lenguaje ensamblador con la facilidad de uso y portabilidad de un lenguaje de alto nivel. UNIX se reprogramó de modo que casi 90% se escribió en C y el resto en lenguaje ensamblador.

En 1985 Bjarne Stroustrup, también de laboratorios Bell, inventó el lenguaje de programación C++.

Añadió al lenguaje C características para abstracción de datos y programación orientada a objetos.

Desde los años en que el doctor Stroustrup inventó C++, el lenguaje comenzó a evolucionar en formas un poco diferentes en distintos compiladores de C++. Aunque en un principio C se propuso como un lenguaje de programación de sistema, tanto C como C++ se emplean mucho hoy día en negocios, industria y computación personal. C++ es poderoso y versátil e incorpora una amplia variedad de conceptos de programación.

1. No aplica.
2. Toda profesión opera con un conjunto de principios éticos que ayudan a definir las responsabilidades de las personas que la practican. Toda profesión opera con un conjunto de principios éticos que ayudan a definir las responsabilidades de las personas que la practican. El conocimiento puede ser empleado también de maneras no éticas. Una computadora puede ser programada para activar una bomba, sabotear la línea de producción de un competidor o robar dinero.

Es fácil copiar el software de computadora. Pero está en general protegido por las leyes de los derechos de autor. Es ilegal copiar software sin permiso de su creador. Los profesionales de la computación tienen la obligación ética de no participar en la piratería de software y deben tratar de evitar que ocurra.

1. Hay esencialmente tres etapas en la escritura de un programa:

**Fase de resolución del problema**

1. Análisis y especificación. Entender (definir) el problema y lo que debe hacer la solución.

2. Solución general (algoritmo). Desarrollar una secuencia lógica de pasos que resuelve el problema.

3. Verificar. Seguir los pasos exactamente para ver si la solución resuelve en realidad el problema.

**Fase de implementación**

1. Solución concreta (programa). Traducir el algoritmo en un lenguaje de programación.

2. Prueba. Ver que la computadora siga las instrucciones. Después, comprobar de manera manual

los resultados. Si encuentra errores, analice el programa y el algoritmo para determinar la fuente

de errores, y luego hacer correcciones.

Una vez que se ha escrito el programa, entra a la tercera fase: mantenimiento.

**Fase de mantenimiento**

1. Uso. Utilizar el programa.

2. Mantenimiento. Modificar el programa para satisfacer requisitos de cambio o corregir cualquier error que aparezca al usarlo.

1. Un compilador es un programa que traduce los programas escritos en algunos lenguajes de alto nivel (C++, Pascal, FORTRAN, COBOL, Modula-2 y Ada, por ejemplo) en lenguaje de máquina.
2. Hardware: Componentes físicos de una computadora.

Software: Programas de computadora; conjunto de programas disponibles en una computadora.

1. Entiendo que la secuencia descrita en el punto 8 da un método apropiado de resolución de problemas para desarrollar una solución algorítmica del mismo. Más aún: haría especial hincapié en el punto 1 de la primera fase, ya que desarrollar una solución general antes de escribir un programa ayuda a manejarlo, mantener claros los pensamientos y evitar errores. Si al comienzo no se da tiempo para razonar y pulir el algoritmo, seguramente se utilizará tiempo extra en la depuración y revisión del programa. Así que es mejor pensar primero y codificar después. Probablemente mientras más pronto empecemos a codificar, más tiempo nos llevará elaborar un programa que funcione.