**Instituto Superior de Formación Técnica Nº 151**  **Carrera: Analista en Sistemas  
1 Año. Algoritmos y Estructuras de Datos.**

|  |  |
| --- | --- |
| **Trabajo Práctico Nº0** | **Unidad 0** |
| **Modalidad:** Semi -Presencial | **Estratégica Didáctica:** Trabajo individual. |
| **Metodología de Desarrollo:** Det. docente | **Metodología de Corrección:** Via Classroom. |
| **Carácter de Trabajo:** Obligatorio – Con Nota | **Fecha Entrega:** A confirmar por el Docente. |

**Marco Teórico:**  
Responder el siguiente cuestionario en función de la bibliografía Obligatoria.  
Bienvenidos a este Tramo formativo.  
Usaremos la Plataforma como medio de Conexión "Oficial" para poder avanzar desde lo Académico mientras dure la emergencia, luego será un vínculo más para poder estar más interconectados, sirviendo de "Soporte" a las clases presenciales.  
Comenzaremos con tareas de Lectura, Comprensión y resolución de Consignas sobre los temas propuestos a modo de Inducción en la materia.  
  
Por Favor Leer el Material y Responder las Siguientes consignas.  
  
1. Definir el concepto de programa de computadora.  
2. Que 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 harreare y software.  
11. Describir un método apropiado de resolución de problemas para desarrollar una solución algorítmica a un problema.  
  
Utilizar el Libro "Programación-y-resolución-de-problemas-con-c++" Cap. 1  
Bibliografia - Bjarne Stroustrup - El lenguaje de programación C++  
Que será el material "Obligado" de este año.  
  
.

La Cátedra.   
Lic. Oemig José Luis.

1) Programa de computadora: secuencia de instrucciones que entiende la computadora y que persigue un objetivo: resolver problemas.

2) Algoritmo: Son un conjunto de instrucciones sistemáticas y previamente definidas, están ordenadas y acotadas a manera de pasos a seguir para alcanzar un objetivo en un tiempo finito. Abarcan desde una receta de cocina hasta el motor de búsqueda de un sitio web.

3) Un lenguaje de programación es una forma de comunicarnos con una computadora u otro dispositivo e indicarle que hacer.

Existen distintos tipos de lenguaje: principalmente de bajo nivel y de alto nivel. La diferencia se encuentra en lo cerca o lejos que estemos del hardware de nuestro equipo. Esta cercanía tiene que ver con el control que tengamos sobre el dispositivo, placa o controlador.

Lenguaje de alto nivel : es el que se encuentra más cerca del lenguaje natural (en especial para los angloparlantes), operaciones matemáticas conocidas, es independiente de la máquina: es a su vez el más alejado del lenguaje de la computadora. Ejemplos de lenguajes de alto nivel son C++, FORTRAN, Python, Javascipt, etc.

4) **Compilación:** es un proceso que traduce los programas escritos en alto nivel a un código de máquina.

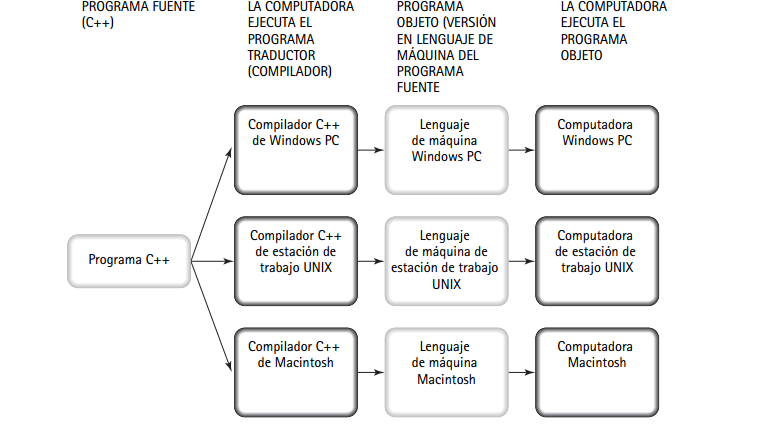
Transforma un lenguaje fuente en un programa objeto, es decir, una versión en lenguaje de bajo nivel.

Obtiene dos salidas:

1- Listado del programa con mensajes de error

2- Programa objeto

**Ejecución**: es la carga del programa objeto en la memoria del ordenador y reemplaza al compilador. De esta manera el CPU ejecuta el programa objeto y las instrucciones del programa.



5) Breve historia de C++

El lenguaje C++ se desarrolló al igual que su antecesor, C, dentro los Laboratorios Bell de AT&T, su creador es el danés [Bjarne Stroustrup, quien comenzó a desarrollarlo a fines de la década del 70 y su primer lanzamiento comercial fue en 1985.](https://es.wikipedia.org/wiki/Bjarne_Stroustrup)

Stroustrup se basó en el lenguaje C el cual Dennis Ritchie inventó 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.

Después, en 1979, Bjarne Stroustrup, también empleado de Bell AT & T, comenzó a trabajar en el lenguaje C con las clases. Tomó prestadas las características deseables de muchos otros lenguajes como Simula, Ada, ML, CLU y ALGOL 68. Por lo tanto, además de las características del lenguaje C, C ++ también incluía clases, verificación de tipo fuerte, argumento de función predeterminada y herencia básica. Hasta 1983, se llamaba C con clases, y en 1983 se llamaba C ++. Durante 1998, un comité conjunto ANSI-ISO publicó la especificación para los estándares de lenguaje C ++.

Fascinado por el enfoque orientado a objetos Stroustrup pensó en implementar este paradigma en el desarrollo de software, sin embargo, el lenguaje Simula era demasiado lento para el uso práctico. Así que comenzó a trabajar en C con clases, es decir, comenzó a trabajar en un nuevo lenguaje que tendría un paradigma orientado a objetos mezclado con las características del lenguaje de programación C.

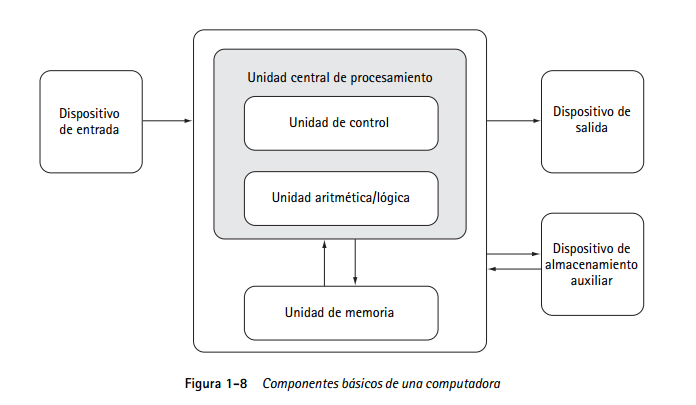
En 1983, se denominó C ++ e incluía algunas características complementarias, como clases, herencia, alineación, argumentos de funciones predeterminadas, polimorfismo, encapsulación y verificación de tipos fuerte.

La primera edición comercial del lenguaje de programación C ++ se lanzó en octubre de 1985.

6) Los componentes principales de una computadora con dispositivos de entrada y salida unidad de procesamiento central y dos tipos de memoria.

La parte de la computadora que sigue las instrucciones se llama unidad central de procesamiento (CPU). Por lo común, el CPU tiene dos componentes. La unidad aritmética/lógica (ALU) efectúa las operaciones aritméticas (suma, resta, multiplicación y división) y las operaciones lógicas (comparar dos valores). La unidad de control regula las acciones de los otros componentes de la computadora para que las instrucciones del programa se ejecuten en el orden correcto.

Los dispositivos de entrada/salida son los que permiten que entren y salgan los datos



7) La

7)Importancia de los asuntos éticos

Toda profesión opera con un conjunto de principios éticos que ayudan a definir las responsabilidades de las personas que la practican.

El conocimiento de las computadoras puede ser un medio poderoso para efectuar un cambio positivo. El conocimiento puede ser empleado también de maneras no éticas.

Es fundamental saber que no existe una neutralidad en el desarrollo de una disciplina sino que puede haber consecuencias a partir de uso y no sólo algo extremo como el desarrollo de una bomba, un error puede tener consecuencias negativas para el usuario final, la empresa o quien dependa de nuestro trabajo, puede fallar un avión, alguna clase de juegos que genere adicción en las personas.

Existen otras cuestiones éticas relacionadas a al piratería, al uso de programas y sistemas operativos copiados cuando requieren un pago por su licencia; la obtención de datos sin consentimiento; el uso de recursos de una computadora sin permiso del dueño, entre otras cuestiones.

8) Etapas básicas en el desarrollo 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.

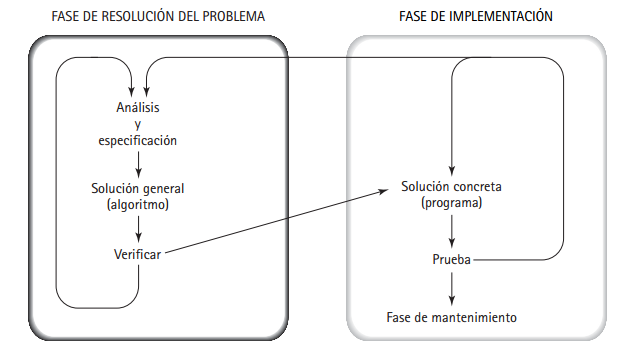
Una vez que se ha escrito el programa, entra a la tercera fase: mantenimiento.

**Fase de mantenimiento**

**1. Uso**. Utilice el programa.

**2. Mantenimiento**. Modifique el programa para satisfacer requisitos de cambio o corregir cual-

quier error que aparezca al usarlo.



9) Compilador Un compilador es un programa informático que traduce todo el código fuente de un proyecto de software a código máquina antes de ejecutarlo. Solo entonces el procesador ejecuta el software, obteniendo todas las instrucciones en código máquina antes de comenzar. De esta manera, el procesador cuenta con todos los componentes necesarios para ejecutar el software, procesar las entradas y generar los resultados. No obstante, en muchos casos, durante el proceso de compilación tiene lugar un paso intermedio fundamental: antes de generar la traducción final en código máquina, la mayoría de los compiladores suelen convertir el código fuente en un código intermedio (también llamado código objeto) que, a menudo, es compatible con diversas plataformas y que, además, también puede ser utilizado por un intérprete.

Al producir el código, el compilador determina qué instrucciones van a enviarse al procesador y en qué orden. Si las instrucciones no son interdependientes, incluso es posible que puedan procesarse en paralelo.

Entre los lenguajes compilados puros se incluyen, entre otros, los veteranos C, C++ y Pascal.

10) Hardware: hacemos referencia a la parte física y tangible de los dispositivos digitales. Se trata del soporte palpable, que permite que los programas se ejecuten y se visualicen, entre otras muchas funcionalidades. Está formado por una gran variedad de componentes, desde los circuitos integrados, los microprocesadores, hasta la pantalla, el teclado o las unidades de almacenamiento.

Software: es la parte intangible de la tecnología. Está compuesto por líneas de código y algoritmos que proporcionan indicaciones a los componentes de hardware. En el contexto de la informática, por tanto, este término se refiere a los programas, aplicaciones y datos que se ejecutan. Según su funcionalidad, destaca el software de sistema y el software de aplicación. El primero es esencial para las operaciones básicas del hardware y la gestión de recursos. El segundo tipo es imprescindible para la realización de tareas concretas.

11) Como crear un algoritmo a partir de un problema

-Haga preguntas

Si se le encomienda una tarea de manera verbal, usted hace preguntas —¿cuándo?, ¿por qué?, ¿dónde?—, hasta que entiende exactamente lo que tiene que hacer. Si las instrucciones son escritas, podría poner interrogaciones en el margen, subrayar una palabra u oración, o indicar de alguna otra manera que la tarea no está clara. Sus preguntas pueden ser contestadas en un párrafo posterior, o bien podría tener que discutirlas con la persona que le asignó la tarea.

-Busque cosas que sean familiares

Nunca reinvente la rueda. Si existe una solución, utilícela. Si ha resuelto antes el mismo problema o uno similar, sólo repita su solución.

En la programación, algunos problemas ocurren una y otra vez en modos distintos. Un buen programador reconoce de inmediato una subtarea que ha resuelto antes y la inserta en la solución.

-Resuelva por analogía

Con frecuencia un problema le recuerda otro similar que ha visto antes. Puede tener a mano la solución de un problema con más facilidad si recuerda cómo resolvió el anterior. En otras palabras, haga una analogía entre los dos problemas. La analogía es sólo una aplicación más amplia de la estrategia de buscar cosas que son familiares. Cuando intenta hallar un algoritmo para resolver un problema, no se limite a las soluciones orientadas a la computadora. Retroceda e intente obtener una visión más amplia del problema. No se preocupe si su analogía no tiene una correspondencia perfecta; la única razón para usar una analogía es que da un lugar para empezar.

-Análisis de medios y fines

Con frecuencia se dan el lugar inicial y el lugar final; el problema es definir un conjunto de acciones que se puedan usar para ir de uno al otro.

Una vez que ha reducido el conjunto de acciones, tiene que resolver los detalles. Podría ser útil establecer objetivos intermedios que son más fáciles de satisfacer que el objetivo global. La estrategia global de análisis de medios y fines es definir los fines y luego analizar los medios para lograrlos. El proceso se traduce con facilidad a la programación de computadoras. Se comienza por escribir lo que es la entrada y lo que debe ser la salida. Luego, se consideran las acciones que puede realizar una computadora y se elige una secuencia de acciones que pueden transformar los datos en resultados.

-Dividir y vencer

Con frecuencia los problemas grandes se dividen en unidades más pequeñas que son más fáciles de manejar. Limpiar toda la casa podría parecer abrumador; limpiar las habitaciones una a la vez parece mucho más accesible. El mismo principio se aplica a la programación. Se descompone un problema grande en piezas más pequeñas que se pueden resolver de manera individual.

-Método de bloques de construcción

Otra forma de enfrentar un problema grande es ver si hay soluciones para las piezas más pequeñas del problema. Sería posible juntar algunas de estas soluciones para resolver la mayor parte del problema grande. Esta estrategia es sólo una combinación de los métodos “buscar cosas familiares” y “divide y vencerás”. Examine el problema grande y vea que se puede dividir en problemas más pequeños para los cuales ya existen soluciones. Resolver el problema grande es sólo una manera de agrupar las soluciones existentes, como pegar bloques o ladrillos para construir una pared.

-Combinar soluciones

Otra forma para combinar las soluciones existentes es agruparlas en una base paso a paso. Por ejemplo, para calcular el promedio de una lista de valores, se debe sumar y contar los valores. Si ya se tienen soluciones separadas para sumar valores y para contarlos, es posible combinarlos. Pero si primero se hace la suma y luego se cuenta, se tiene que leer la lista dos veces. Se pueden ahorrar pasos si se combinan estas dos soluciones: leer un valor y luego sumarlo al total corriente y sumar 1 a la cuenta antes de ir al siguiente valor. Siempre que las soluciones a los subproblemas dupliquen los pasos, piense en agruparlas en lugar de unirlas extremo con extremo.

-Bloqueos mentales: el temor de empezar

La mayoría de los bloqueos mentales se deben a que no se entiende el problema. Reescribir el problema con sus propias palabras es una buena forma de centrarse en las subpartes que lo componen, una a la vez, y entender lo que se requiere para una solución.

-Resolución algorítmica de problemas

Sugerir un procedimiento paso a paso para resolver un problema particular no es siempre un proceso predeterminado. De hecho, normalmente es un proceso de prueba y error que requiere varios intentos y refinamientos. Se prueba cada intento para ver si en realidad resuelve el problema. Si es así, muy bien. De lo contrario, se intenta de nuevo. Resolver cualquier problema no trivial requiere, por lo común, una combinación de técnicas ya descritas.

12. Describir el concepto que afirma “SOMMERVILLE” en su libro Ingeniería de Software, un

modelo del proceso del software es una representación abstracta de un proceso del

software.(leer material complementario)

13. Describir el Modelo Iterativo e Incremental

14. Relacionar UML, Ingeniería de Software y modelos.