Laboratorio de Computación I

Unidad 1

Unidad 1: Introducción a C

Introducción. Estructura de un programa en C. La directiva #include. La directiva #define. Tipos de Datos: Constantes, variables. Operadores matemáticos, relacionales y lógicos. Precedencia entre operadores. Operadores de asignación.Funciones básicas de entrada/salida: printf, scanf, getchar, putchar.

Unidad 1: Objetivos

- Que el alumno comprenda el papel de la sintaxis y de la semántica en la codificación de programas.
- Que el alumno caracterice e identifique los elementos de un Programa en C.
- Que el alumno identifique el proceso desde la codificación hasta la ejecución de un programa.
- Que el alumno adquiera destreza en la codificación de programas sencillos en C.
- Que el alumno identifique y caracterice los distintos tipos de datos.
- Que el alumno identifique las funciones básicas de entrada y salida.
- Que el alumno identifique y conozca la precedencia de operadores
- Que el alumno comprenda los distintos mensajes de error para su posterior corrección.

Concepto de Programación

Consiste en la elaboración de programas para la resolución de problemas mediante computadoras.

El programador se encarga de escribir, probar, depurar y mantener el código fuente.

La programación se realiza mediante el uso de **algoritmos**, que son secuencias finitas, ordenadas y no ambiguas de instrucciones que deben seguirse para resolver un problema.

Lenguaje de Programación

Un Programa de computadora, es una colección de instrucciones que, al ser ejecutadas por el CPU de una máquina, llevan a cabo una tarea o función específica.

Tipos de Lenguaje de Programación

La clasificación más común es:

- **Lenguaje de Bajo Nivel**
- **Lenguaje de Alto Nivel**

Tipos de Lenguaje de Programación

Se pueden utilizar muchos lenguajes para programar una computadora.

- **Lenguaje de Bajo Nivel:**

Consiste el una colección de instrucciones que controlan el hardware de la computadora.

- Es el más básico. fueron los primeros que surgieron y se llaman así porque están directamente relacionados con el hardware del computador Dependen totalmente de la máquina y no se pueden utilizar en otras máquinas
- Estos lenguajes son los que ordenan a la máquina operaciones fundamentales para que pueda funcionar, por ejemplo la asignación y liberación de memoria, el uso de punteros, la creación de tipos de datos, etc.

Tipos de Lenguaje de Programación

Se pueden utilizar muchos lenguajes para programar una computadora.

- **Lenguaje de Bajo Nivel:**
  - El primer lenguaje de este tipo es el **Lenguaje de Máquina.**
  - **Lenguaje de Máquina:** Consiste en una serie de instrucciones en binario (ceros y unos).
  - Este lenguaje es muy complicado y la posibilidad de cometer errores es muy alta, para resolver esta dificultad surgió el Lenguaje Ensamblador.
  - **Lenguaje Ensamblador:** Consiste en asignar una abreviatura a cada instrucción en binario de tal forma que sea más fácil recordarla y más difícil equivocarse. Continua siendo necesario conocer el hardware de la computadora.

Tipos de Lenguaje de Programación

- **Lenguajes de alto nivel:**
  - Las instrucciones que utilizan son más compatibles con los lenguajes y la forma de pensar humanos.
  - La mayoría son lenguajes de propósito general, como C, Pascal, Java, PHP,etc.
  - No dependen de la máquina y sirven fundamentalmente para crear programas informáticos que solucionan diferentes problemas.
  - Son los más usados por los programadores y por todo del mundo que realiza programas informáticos.
  - Pero aunque el programador de esta forma se distancie del hardware del computador, este sigue trabajando en lenguaje máquina. Por ello se hace necesaria una traducción a una secuencia de instrucciones interpretables por el computador. ***Esta labor es llevada a cabo por los compiladores y los intérpretes.***

Tipos de Lenguaje de Programación

- **Compiladores**

Son aquellos cuya función es traducir un programa escrito en un determinado lenguaje a un idioma que la computadora entienda (lenguaje máquina con código binario).

- Al usar un lenguaje compilado (como lo son los lenguajes del popular Visual Studio de Microsoft), el programa desarrollado nunca se ejecuta mientras haya errores, sino hasta que luego de haber compilado el programa, ya no aparecen errores en el código.

Tipos de Lenguaje de Programación

- **Intérprete**

Es aquel programa que analiza el programa fuente y lo ejecuta directamente sin generar ningún código equivalente.

- No se graba el código objeto para utilizarlo posteriormente.
- La siguiente vez que se utilice una instrucción, se le debe interpretar otra vez y traducir a lenguaje máquina. Por ejemplo, durante el procesamiento repetitivo de los pasos de un ciclo, cada instrucción del ciclo tendrá que volver a ser interpretado cada vez que se ejecute el ciclo, lo cual hace que el programa sea más lento en tiempo de ejecución (porque se va revisando el código en tiempo de ejecución) pero más rápido en tiempo de diseño (porque no se tiene que estar compilando a cada momento el código completo).
- El intérprete elimina la necesidad de realizar una corrida de compilación después de cada modificación del programa cuando se quiere agregar funciones o corregir errores;

Pasos para realizar un Lenguaje de Programación

Pasos	Etapa	Descripción
1	Análisis del problema	Conducen al diseño detallado por medio un código escrito en forma de un algoritmo.
2	Diseño de algoritmo	
3	Codificación	Se implementa el algoritmo en un código escrito en un lenguaje de programación. Refleja las ideas desarrolladas en las etapas de análisis y diseño
4	Compilación y ejecución	Traduce el programa fuente a programa en código de maquina y lo ejecuta.
5	Verificación	Busca errores en las etapas anteriores y los elimina.
6	Depuración	
7	Documentación	Son comentarios, etiquetas de texto, que facilitan la comprensión del programa

Sintaxis y Semántica

La **sintaxis** de un lenguaje define como se pueden poner juntos símbolos, palabras reservadas, e identificadores para hacer un programa válido.

La **semántica** es el significado del constructor; ella define su papel en un programa.

Un programa sintácticamente correcto no implica que sea lógicamente (semánticamente) correcto.

Buenas prácticas deseables de un programa

Cuando se codifica una aplicación software, existe un número potencialmente infinito de programas que satisfacen los mismos requisitos.

Sin embargo, no todos estos programas comparten los mismos atributos de calidad.

Es decir, no todos son iguales de eficientes tanto en consumo de procesador como de memoria; no todos son igual de legibles; no todos son igual de fáciles de modificar; no todos son igual de fáciles de probar y verificar que funcionan correctamente, etc.

Buenas prácticas deseables de un programa

No es solamente importante crear aplicaciones software que funcionen correctamente, sino que además debe crear aplicaciones software robustas, eficientes y fáciles de mantener.

Características Deseables de un programa

- Integridad
- Claridad
- Sencillez
- Eficiencia
- Modularidad
- Generalidad

Características Deseables de un programa

Integridad:

Se refiere a la corrección de los cálculos ya que la integridad de los cálculos es absolutamente necesaria en cualquier programa de computadora.

Claridad:

Hace referencia a la facilidad de lectura del programa en conjunto, con particular énfasis en la lógica subyacente. Si un programa está escrito de forma clara, será posible

- para otro programador seguirla lógica del programa sin mucho esfuerzo.
- para al autor original seguir su propio programa después de haberlo dejado durante un periodo de tiempo.

Características Deseables de un programa

Sencillez:

La claridad y corrección de un programa se suelen ver favorecidas con hacer las cosas de forma tan sencilla como sea posible, consistente con los objetivos del programa en su conjunto.

Eficiencia:

Está relacionada con la velocidad de ejecución y la utilización eficiente de la memoria. Éste es uno de los objetivos importantes, aunque no se debe conseguir a expensas de la pérdida de la claridad o la sencillez.

Características Deseables de un programa

Modularidad:

Muchos programas se pueden dividir en pequeñas subtareas. , es decir implementar cada una de estas subtareas como un módulo separado del programa. En C estos módulos son las funciones. Permite aumenta la corrección y claridad de éstos y facilita los posibles cambios futuros del programa.

Generalidad:

Normalmente es deseable que un programa sea lo más general posible, dentro de unos límites razonables. Por ejemplo, podemos hacer un programa que lea los valores de ciertos parámetros en lugar de dejarlos fijos. Como norma general se puede conseguir con muy poco esfuerzo adicional un nivel considerable de generalidad.

Buenas Prácticas de Programación

- Escribir los programas lo más simple y directo posible.
- Todo programa debe ser previamente comentado, explicando el propósito, funcionamiento completo y el resultado esperado.
- Dentro de las funciones definidas, establece un comentario, que resalte la estructura funcional de la aplicación y facilite la lectura al programador al que le corresponda analizar el código.
- Usar un nivel de sangría (indentación) por cada bloque de código (sentencias condicionales y bucles son consideradas como bloques de código embebido dentro de otro, por lo que se recomienda la misma, ésta indentación corresponde a una sangría que comúnmente tiene el valor de una tabulación (tecla Tab) o bien tres o cuatro espacios.

Buenas Prácticas de Programación

- Es importante que el tamaño de las sangrías sean regulares (consistentes) y no varíen a lo largo del código.
- Se recomienda declarar variables en líneas separadas, ya que se facilita la descripción de cada variable mediante comentarios.
- No usar variables cuyo nombre no posea algún significado descriptivo, una variable con nombres significativos permite al lector entender el contexto del código y permite disminuir la cantidad de documentación asociada.
- Evitar el código *commented-out*, que corresponde al código comentado para que no se ejecute/no compile, ya que la lectura del código se vuelve engorrosa.

Buenas Prácticas de Programación

- En caso de usar operadores binarios (por ejemplo +, -, &&, ||, entre otros) se recomienda poner espacio a los extremos de cada operador, de modo que se resalte su presencia y se facilite la lectura del código.
- Evitar la incorporación de más de una instrucción por línea.
- Cuando se escribe operaciones que hagan uso de muchos operadores, procura revisar que las operaciones se estén realizando en el orden que se espera que se realicen.
- Nunca olvidar inicializar los contadores y sumadores.