***Resumen Programación***

**Unidad 1: “Proceso de programación”**

Procesos intelectuales del programador:

1. **Abstracción:** *“Interpretar los aspectos esenciales de un problema y expresarlo en términos precisos”. (cálculos, repeticiones, dependencias, etc.)*
2. **Modelización del problema:** *“Simplificar su expresión, encontrando sus aspectos principales, los datos que se habrán de procesar y el contexto”. (datos necesarios para procesar)*
3. **Especificación del problema real:** *“Determinar en forma clara y concreta el objetivo que se desea”. (hacer funciones, pseudocódigo)*
4. **Expresión de soluciones ejecutables en la PC:** *“Realizar una solución ejecutable en una computadora usando un lenguaje de programación”. (escribir en un lenguaje de programación)*

Etapas de Resolución de Problemas:

**Análisis del Problema:** Importancia del contexto: La definición del contexto es importante para analizar y diseñar la solución usando computadoras. Impone restricciones y consideraciones.

**Diseño de la Solución:** Descomposición / Modularizarían: Se utiliza la metodología “Top-Down” de descomposición de problemas para desarrollar el sistema de software. Se obtendrán módulos que deberán estar ligados entre sí para obtener la solución final.

**Especificación del Algoritmo:** Cada módulo tendrá su propio algoritmo y de la elección del mismo dependerá la eficiencia de la solución.

**Escritura de Programas:** Un algoritmo es una especificación simbólica que debe convertirse a un programa real sobre un lenguaje de programación concreto.

**Verificación:** Antes de finalizar una tarea de programación, es importante preparar datos representativos y realizar pruebas exhaustivas, depuración, considerar alternativas de diseño y aplicar un buen estilo de programación. Esto garantiza un programa más confiable y menos propenso a errores. Durante la verificación pueden surgir diferentes errores:

* **Errores de ejecución**: son condiciones que afectan la operación normal del programa y pueden originarse por múltiples circunstancias.
* **Errores de sintaxis**: son todos aquellos que se generan por no cumplir con las “normas de escritura de un lenguaje.
* **Errores de tipo lógico:** son todos aquellos derivados de un mal diseño de los algoritmos

Hay 3 criterios básicos a tener en cuenta: Correctitud (resultados deseados), Claridad y Eficiencia.

*“Cuanto más exigente sean las pruebas mayor seguridad tendrá el programa”*

**Unidad 2 - I: “Lenguajes de Programación - C”**

**Escritura de programas:**

* El algoritmo se “traduce” al lenguaje de programación elegido.
* Escribir las instrucciones.
* Programar **no es lo mismo** que codificar

**Lenguaje de Programación:** es un lenguaje formal que proporciona una serie de instrucciones que permiten a un programador escribir secuencias de órdenes y algoritmos a modo de controlar el comportamiento físico y lógico de una computadora con el objetivo de que produzca diversas clases de datos.

**Programa:** conjunto de órdenes y datos escritos mediante un lenguaje de programación.

**Lenguajes:**

* Bajo nivel: lenguajes más próximos a la arquitectura hardware
* Alto nivel: lenguajes más cercanos a los programadores y usuarios.

**Lenguaje de Bajo Nivel:**

* Totalmente dependientes de la máquina, el programa que se realiza con este tipo de lenguajes no se puede migrar o utilizar en otras máquinas.
* Aprovecha al máximo las características del hardware
* Lenguaje máquina: este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en combinaciones de 0s y 1s para formar ordenes entendibles por el hardware. Es más rápido pero difícil de manejar.
* Lenguaje ensamblador (Assambly): derivado del lenguaje de máquina formado por abreviaturas de letras y números llamados mnemotécnicos. Este lenguaje necesita traductor, los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria, pero seguía siendo difícil de probar y mantener.

**Lenguaje de Alto Nivel:**

* Se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Están dirigidos a solucionar problemas mediante el uso de Estructuras Dinámicas.
* Lenguajes independientes de la arquitectura del ordenador, es decir, puede migrarse de una maquina a otra.
* Permiten que el programador se olvide por completo del funcionamiento interno de la máquina, solo hace falta un traductor que entienda el código fuente como las características de la máquina.

**Programación estructurada**: es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora. Puede ser secuencia, selección e iteración.

**Características de C:**

* Lenguaje para programación estructurada.
* Es tipificado.
* Contiene muy pocas palabras reservadas.
* No contiene órdenes para trabajar con objetos compuestos.
* Distingue entre mayúsculas y minúsculas.

**Ventajas:**

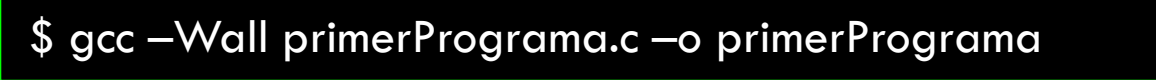
* Tiene muchos compiladores para casi todos los sistemas.
* Facilidad para realizar programas modulares y/o utilizar código o bibliotecas existentes.
* Lenguaje muy flexible, veloz y potente, permite un software efectivo.
* Acceso a memoria de bajo nivel mediante el uso de punteros.

**Desventajas:**

* No tiene instrucciones propias para la asignación dinámica de memoria, ni instrucciones de entrada/salida.
* Todas estas operaciones de alto nivel pueden ser realizadas por funciones explícitamente.
* Se requiere tiempo en conseguir el ejecutable, porque cada vez compila todo el fichero.
* No dispone de sistemas de control automáticos.

**Unidad 2 - II: “Etapas de compilación”**

Compilador: el compilador es un programa informático que traduce un programa de un lenguaje a otro, generando un lenguaje equivalente que la máquina será capaz de interpretar. Este proceso de traducción se conoce como compilación



$ gcc: “Gnu C Compiler”.  
-Wall: active las advertencias del compilador.  
-o: Archivo de salida.  
“primerPrograma.c”: archivo fuente.  
“primerPrograma”: archivo de ejecutable de salida.

**Etapas de compilación**

Cada una tiene un código fuente como entrada y un código objeto como resultado.

**Preprocesamiento:** el preprocesamiento es realizado por el preprocesador. Traduce el archivo fuente que es una forma ampliada del lenguaje.



-E: Detiene la compilación luego del preprocesamiento.

**Compilación:** Transforma el código C preprocesado en el lenguaje ensamblador propio del procesador de nuestra máquina.



-S: Detiene la el proceso luego de la etapa de compilación. La salida será en forma de código Assambly. Por defecto el archivo tiene la extensión “.s”.

**Ensamblado:** Traduce el programa escrito en lenguaje ensamblador a código binario.



-c: realiza las 3 primeras etapas, creando un archivo “.o” que contiene el código binario listo para ser enlazado.

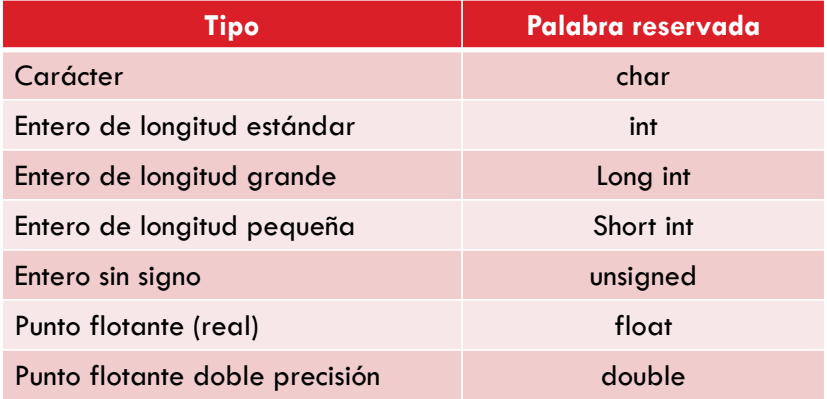
**Enlazado:** Las funciones se encuentran compiladas y ensambladas en librerías existentes. Incorpora el código binario de estas funciones en el lenguaje final.



Linea de comandos para compilar archivos en C.

**Unidad 3 - I: “Tipos de datos”**

**Tipos de Datos:** los tipos de datos determinan que elemento o información contendrá el área de memoria reservada para las variables declaradas. Las palabras claves constituyen sus respectivas especificaciones.



**Variables:** es un objeto cuyo valor cambia durante la ejecución del programa, tiene nombre y ocupa una cierta posición de memoria. Todas las variables que se usan deben ser declaradas.

**Constantes:** una línea **#define** define una constante simbólica



**Función de Salida (prinft):** Permite la presentación de valores numéricos, caracteres y cadenas de texto por el archivo estándar de salida (pantalla).

**Formato de modificadores %:** entre el % y el carácter de conversión puede haber:

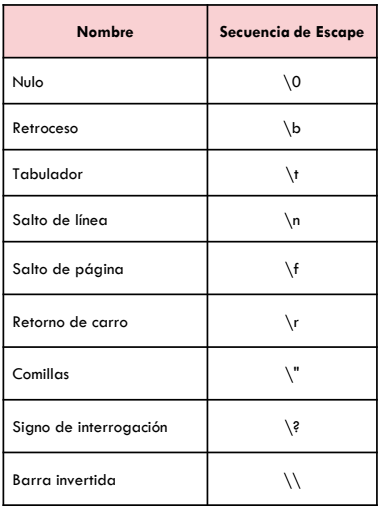
% [signo] [longitud] [.precisión] [l/L] conversión

* **Signo:** “-“ indica que se ajustará a la izquierda o “+” para que se imprima con signo.
* **Longitud:** especifica un ancho mínimo de campo, si tiene menos caracteres que el ancho entonces lo rellena a la izquierda.
* **Precisión:** indica el número máximo de decimales que tendrá el valor
* **I/L:** “I” para variable de tipo long y “L” cuando es de tipo double

**Listado de modificadores:**

* **%c** Un único carácter
* **%d** Un entero con signo, en base decimal
* **%u** Un entero sin signo, en base decimal
* **%o** Un entero en base octal
* **%x** Un entero en base hexadecimal
* **%e** Un número real en coma flotante, con exponente
* **%f** Un número real en coma flotante, sin exponente
* **%s** Una cadena de caracteres
* **%p** Un puntero o dirección de memoria

**Secuencias de Escape**: permiten expresar ciertos caracteres no imprimibles. Las secuencias de escape son interpretadas por el compilador como un solo carácter.



**Función de Entrada (scanf):** Permite ingresar la información o datos en posiciones de memoria de la computadora a través del archivo estándar de entrada (teclado).

**Scanf:** “necesita conocer” la posición de memoria donde se encuentra la variable para poder almacenar información ingresada. Por eso se utilizar un operador unario de dirección **“&”** (ampersand).

**Unidad 3 - II: “Tipos de datos, Operadores y Expresiones”**

Tipo de datos simples: se caracterizan porque sus componentes no se pueden descomponer en tipos más simples. Ejemplo: entero, carácter.

sadsad