Apunte Teórico para el Examen Final de Programación I

1. Conceptos Fundamentales de Programación y Calidad del Software

- **Problema del Mundo Real y su Resolución:**
  La informática es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras. La computadora es una herramienta que ejecuta órdenes y ayuda a realizar tareas repetitivas en menor tiempo y con mayor exactitud.
- **Etapas para la Resolución de un Problema:**
  1. **Análisis:** Sintetizar requerimientos, crear un modelo simplificado del problema, analizar entradas y salidas.
  2. **Diseño:** Diseñar una solución, utilizando la descomposición funcional (Dividir y Vencerás) para reducir la complejidad, distribuir el trabajo y permitir la reutilización de módulos.
  3. **Implementación:** Escribir algoritmos en un lenguaje de programación y elegir la representación de datos.
  4. **Verificación:** Estudiar si el sistema cumple con los requerimientos especificados.
- **Definiciones Clave:**
  - **Algoritmo:** Especificación rigurosa de una secuencia finita de pasos para alcanzar un resultado deseado en tiempo finito. Debe ser claro y unívoco.
  - **Dato:** Representación de un objeto, aspecto o situación del mundo real para modelizar el problema. Pueden ser constantes (valor fijo) o variables (valor variable).
  - **Programa:** Algoritmo + Datos. Conjunto de instrucciones u órdenes ejecutables sobre una computadora.
- **Calidad de los Programas:**
  Un programa debe ser **correcto** (cumplir con la función especificada y requerimientos) y **eficiente** (utilizar óptimamente los recursos como memoria y tiempo de ejecución).
  - **Corrección:**
    - **Verificación:** ¿Estamos construyendo el producto correctamente? Comprueba que el software concuerda con su especificación.
    - **Validación:** ¿Estamos construyendo el producto correcto? Asegura que el software cumple las expectativas del cliente.
    - **Técnicas:** Testing, Debugging, Walkthrough.
  - **Legibilidad:** El código fuente debe ser fácil de leer y entender, incluyendo comentarios adecuados. Se vincula con la documentación.

2. Modularización

- **Concepto de Modularización:**
  Estrategia que implica dividir un problema grande en partes funcionalmente independientes, que encapsulan operaciones y datos.
  - **Alta Cohesión:** Grado de identificación de un módulo con una función concreta.
  - **Bajo Acoplamiento:** Medida de interacción entre módulos (baja interacción deseable).
- **Ventajas:**
  - Mayor productividad.
  - Reusabilidad.
  - Facilidad de mantenimiento correctivo.
  - Facilidad de crecimiento del sistema.
  - Mayor legibilidad.
- **Tipos de Módulos en Pascal:**
  - **PROCEDURE (Procedimiento):**
    - Realiza una tarea específica.
    - Puede retornar cero, uno o más valores.

        * El flujo de control retorna a la instrucción siguiente a su invocación.
    - **FUNCTION (Función):**
        * Realiza una tarea específica.
        * Retorna un único valor de tipo simple.
        * Flujo de control retorna a la instrucción de invocación.
        * Pueden pensarse como operadores definidos por el usuario.
- **Comunicación entre Módulos:**
    - Parámetros por Valor (IN)
    - Parámetros por Referencia (VAR)
    - Reglas de Coincidencia: número y tipo de parámetros actuales deben coincidir con los formales.
    - Variables Globales: se desaconseja su uso.
- **Alcance de Variables:**
    - **Locales:** Declaradas dentro de un módulo.
    - **Globales:** Declaradas en el programa principal, accesibles desde cualquier parte.

3. Tipos de Datos Definidos por el Usuario

- **Concepto:** Conjunto de valores permitidos y operaciones sobre ellos.
- **Ventajas:** Expresividad, seguridad, límites, flexibilidad, documentación.
- **Subrango:** Valores consecutivos de un tipo ordinal.
- **String:** Secuencia de caracteres, máxima longitud 255 en Pascal.
- **Conjuntos (SET):**
    - Colección homogénea, sin repetición, sin orden.
    - Operaciones: Unión (+), Intersección (*), Diferencia (-), Pertenencia (IN), Comparación (A<>B, A<=B, A>=B).

4. Estructuras de Datos

- **Concepto:** Conjunto de variables relacionadas que se operan como un todo.
- **Clasificación:**
    - Homogéneas / Heterogéneas
    - Estáticas / Dinámicas
    - Acceso Directo / Secuencial
    - Lineales / No Lineales
- **Registros (RECORD):**
    - Heterogéneos y estáticos.
    - Acceso directo por nombre: `miVariable.campo1`.
    - Operaciones: Asignación, lectura/escritura campo a campo, comparación campo a campo.
- **Arreglos / Vectores (ARRAY):**
    - Colección homogénea y estática.
    - Acceso directo mediante índice.
    - Operaciones: carga, recorrido, agregar al final, insertar, borrar, búsqueda (lineal, optimizada, binaria).
- **Matrices (Array Bidimensionales):**
    - Estructura homogénea, acceso directo por fila y columna.
    - Operaciones: carga, impresión, búsqueda, eliminación, suma, producto.
- **Punteros (^) y Alocación Dinámica:**
    - Almacenan dirección de otra variable.
    - Operaciones: `new()`, `dispose()`, acceso a variable apuntada `puntero^`.
    - Por valor vs por referencia (VAR) como parámetros.
- **Listas Simples Enlazadas:**
    - Nodos con datos y puntero al siguiente.
    - Operaciones: crear lista, agregar, recorrer, buscar, eliminar, insertar ordenado.
- **Listas Dobles Enlazadas:**

  - Nodos con puntero anterior y siguiente.
  - Recorrido bidireccional, inserción/eliminación más compleja.
  - Operaciones: creación, impresión, agregar, insertar ordenado, eliminar por valor.
- **Merge de Listas (ordenadas):**
  - Combinar dos listas ordenadas en una nueva lista ordenada.
  - Pasar listas por referencia para avanzar punteros.

5. Eficiencia de Programas

- **Concepto:** Utilización óptima de recursos (tiempo y memoria).
- **Factores que afectan eficiencia:** datos de entrada, calidad del código compilado, rapidez de instrucciones.
- **Medición:**
  - **Empírico:** Ejecutar el programa.
  - **Teórico:** Contar operaciones elementales.
    - IF-THEN-ELSE, FOR, WHILE, REPEAT-UNTIL.
- **Cálculo de Memoria:**
  - **Estática:** Antes de la ejecución.
  - **Dinámica:** Reserva y liberación durante ejecución.