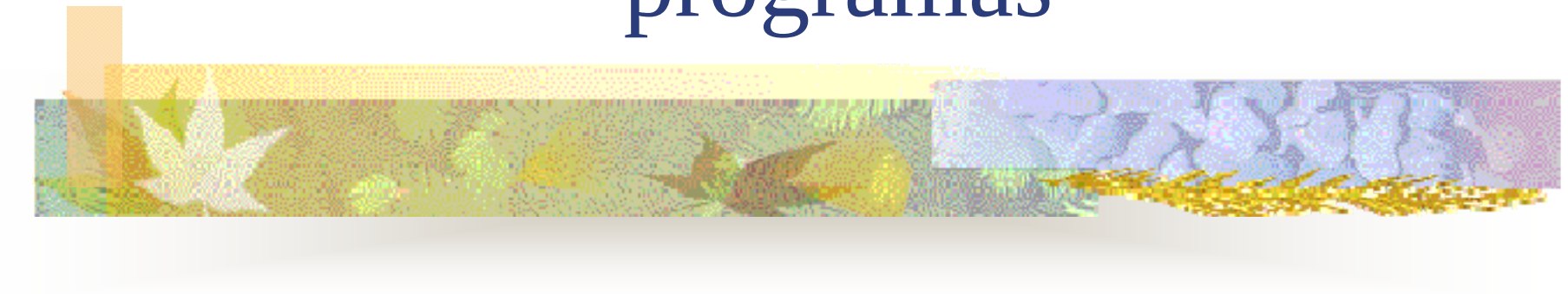
Así habló el maestro programador:
"Cuando el programa está siendo probado, es muy tarde para hacer cambios de diseño."

El Tao de la programación

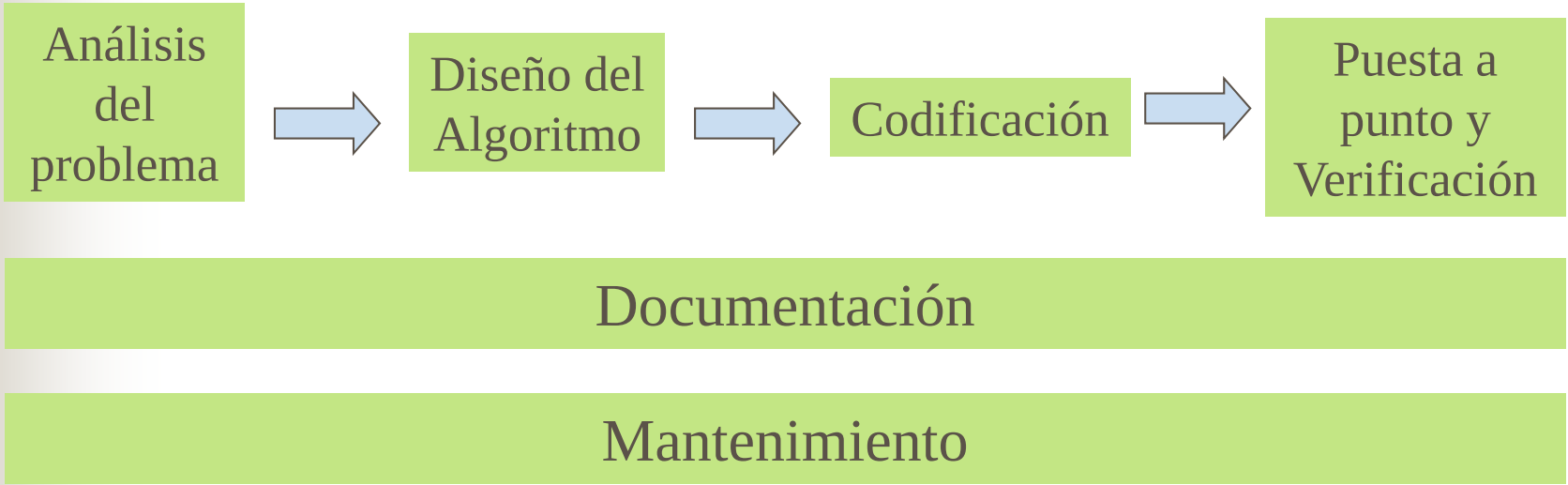
Técnicas para la prueba de programas



Eva Lucrecia Gibaja Galindo
Dpto. Informática y Análisis Numérico

Puesta a punto de un programa

- La puesta a punto y verificación conforman la última fase del proceso de la programación.



Puesta a punto de un programa

- El **objetivo** fundamental es evitar la aparición de errores cuando el programa empiece a funcionar.

- Consiste en localizar, comprobar y corregir los errores de programación.
- **Fases** de la puesta a punto:
  - Detección de errores.
  - Depuración de errores:
    - Localización.
    - Eliminación.
  - Prueba del programa

Puesta a punto de un programa

- Un programa puede poseer distintos **tipos de errores**:
  - **Errores léxicos** (detección en tiempo de compilación).
    - Una palabra clave mal escrita
    - Un identificador con un carácter no autorizado
    - Un operador relacional mal escrito.
  - **Errores sintácticos** (detección en tiempo de compilación).
    - Una llamada a un subprograma con un número de parámetros reales diferente al número de parámetros formales.
    - La utilización de un operador sobre uno o varios operandos incompatibles.
    - Una sentencia de control mal escrita o incompleta.
  - **Errores lógicos** (detección en tiempo de ejecución).
    - Asignación a un objeto de un valor de un tipo diferente al suyo.
    - Utilización de una componente de un vector fuera de rango o inexistente.
    - Ejecución de un bucle infinito.

Prueba del programa

- Objetivos.
  - El objetivo de la prueba es garantizar que el programa funciona y produce los resultados esperados independientemente de la naturaleza de los datos de entrada.

  - Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
  - Una prueba tiene éxito si descubre un error no detectado hasta entonces.
  - La prueba del programa detecta errores pero **NO** garantiza la ausencia de éstos.
- Necesidad de su uso.
  - Es vital en programas cuya incorrección puede provocar daños graves (control del espacio aéreo, control en una central nuclear, etc).
  - El coste de la prueba ha de ser proporcional a la importancia del programa.
  - Para que la prueba sea objetiva, debe realizarse por una persona distinta de la que la construyó.

Prueba del programa

- Definición.
  - Consiste en la aplicación sistemática de ***casos de prueba*** al programa con el objetivo de detectar y corregir los errores o fallos del mismo.
  - Los ***casos de prueba*** consisten en distintos conjuntos de valores de entrada, **elegidos intencionadamente**, para analizar el comportamiento del programa bajo unas condiciones que pueden o no haber sido sugeridas por el diseñador del programa.
  - Se denomina ***juego de pruebas*** al conjunto de todos los datos que están incluidos en estas pruebas.

Programa correcto,robusto,amigable

- Un programa es **correcto** si para cualquier entrada que satisfaga la precondición termina generando unos resultados que satisfacen la postcondición. El mismo concepto se puede aplicar a una función o procedimiento.
- Un programa es **robusto** si:
  - Es correcto.
  - Y para todas las entradas que no satisfacen la precondición, el programa termina y refleja el hecho de que hay un error en la entrada.
- Un programa es **amigable** si:
  - Es robusto.
  - Y además ofrece la posibilidad de que se corrijan los datos de entrada cuando no satisfagan la precondición.

Precondición: conjunto de restricciones que han de cumplir los datos de entrada.

Postcondición: conjunto de restricciones que cumplen los resultados si se cumple la precondición.

El conductor de pruebas

- **El conductor de pruebas** es un programa que guía el proceso mediante el cual se realizan las pruebas.
- El conductor de pruebas puede probar:
  - Un subprograma.
  - Un programa completo.

Prueba de subprogramas

- Prueba de un subprograma:
  - Lee los datos de entrada de un fichero (para evitar errores humanos al teclear y reducir el tiempo de introducción de los datos).
  - Ejecutar el programa.
  - Escribe los datos de entrada y salida
- Un subprograma se prueba con una amplia gama de los valores de entrada y se comprueba que los resultados son consistentes con las postcondiciones.
  - Se selecciona un grupo significativo de casos de prueba, ya que es imposible seleccionarlos todos.

Prueba de un programa completo

- Prueba de un programa completo. El programa se convierte en el programa conductor, encargándose el de leer los datos de entrada del fichero y utilizarlos para producir los datos de salida correspondiente.
- Para probar un programa se comprueban antes los subprogramas que lo componen, y después se prueba el programa completo usando un mecanismo similar. (**Prueba de abajo a arriba**).

Diseño de casos de prueba

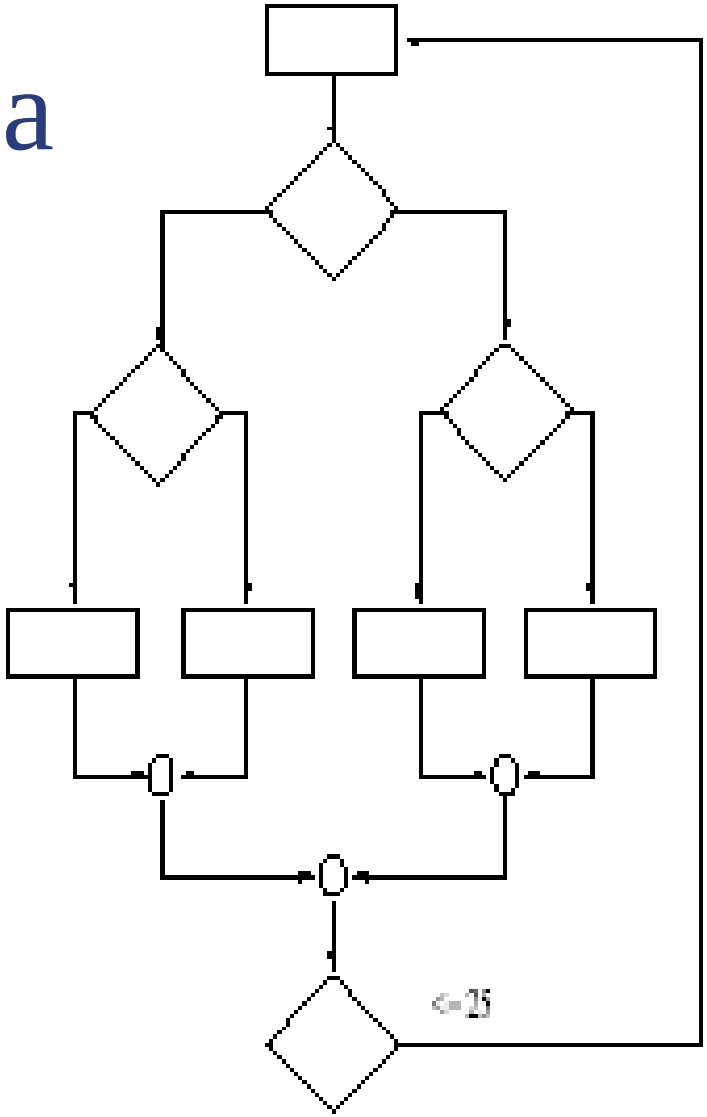
- Pretende obtener los datos que compondrán el conjunto de los casos de prueba.
- Se eligen los datos para que tengan la mayor probabilidad de detectar fallos minimizando esfuerzo y tiempo.
- Técnicas para el diseño de pruebas:
  - **Prueba de la caja negra**: comprueba la corrección del programa en función de sus resultados sin analizarlo internamente.
  - **Prueba de la caja blanca**: comprueba la corrección del programa analizando internamente sus componentes.
- La prueba de caja blanca es considerada como una prueba a pequeña escala, mientras que la prueba de caja negra se considera una prueba a gran escala.
- Se suelen combinar pruebas de caja negra y de caja blanca para comprobar tanto el funcionamiento de interfaz del programa como de sus componentes internos.

Pruebas de caja blanca

- Los objetivos son:
  - Recorrer al menos una vez todos los caminos independientes del programa.
  - Ejercitar todas las decisiones lógicas en sus dos vertientes (verdadera y falsa).
  - Ejecutar todos los esquemas iterativos en sus límites.
  - Observar el correcto funcionamiento de las estructuras de datos.
- Se basan en un **minucioso examen** de los detalles internos del programa.
- Una prueba exhaustiva es generalmente inviable, incluso para programas pequeños. No obstante se puede elegir una serie de caminos lógicos independientes que ejerciten las sentencias y estructuras de datos más importantes del programa.

Pruebas de caja blanca

- Bucle que se ha de ejecutar no mas de 25 veces.
- Cada iteración del bucle puede realizarse por cuatro caminos lógicos independientes.
- Esto implica que para las posibles iteraciones del bucle será necesario probar 4^{25} ($=1.1258999*10^{15}$) posibles caminos lógicos.
- Un procesador que tarde un milisegundo en probar cada caso de prueba tardará más de 35.702 años en realizar la prueba.
- Posteriormente habría que analizar los resultados



Pruebas de caja blanca

- **Prueba del camino básico**.
  - El método del camino básico permite al diseñador obtener un conjunto básico de caminos de ejecución que garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.
- **Prueba de las condiciones**.
  - Permite obtener un conjunto de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa para detectar errores:
    - En un operador lógico, en un paréntesis lógico, en un operador relacional, en una expresión aritmética.
- **Prueba de flujo de datos**.
  - Selecciona caminos de prueba para comprobar que cada variable se comporta correctamente dentro del ámbito asociado a su declaración.
- **Prueba de bucles**.
  - Se centra en la validez de las construcciones de bucles.

Pruebas de caja negra

- No es una alternativa a las pruebas de caja blanca, es un complemento.
- Las pruebas se llevan a cabo sobre la interfaz del programa.
- Los casos de prueba pretenden demostrar que:
  - Los subprogramas son operativos
  - Las entradas se aceptan de forma adecuada
  - Se genera una salida correcta
  - La integridad de la información externa (ficheros) se mantiene.
- Se suelen realizar después de la prueba de caja blanca.
- Pretende encontrar errores del tipo:
  - Funciones incorrectas.
  - Errores de interfaz.
  - Errores en estructuras de datos o en accesos a bases de datos.
  - Errores de rendimiento.
  - Errores de inicialización o terminación