Slide 1

Índice

Slide 2

Plan de trabajo

1. Análisis

   Especificación y batería de pruebas
2. Pruebas de escritorio
3. Diseño

   Revisión y ampliación de batería de pruebas
4. Pruebas de escritorio (trazas, . . .)
5. Codificación

   Revisión y ampliación de batería de pruebas
6. Pruebas de ejecución

Slide 3

ANÁLISIS Y ESPECIFICACIÓN

Entradas:
Salidas:
Objetivo:
Método:
Suposiciones:
Entorno local:
Constantes:
Variables:
Usa:
...
Batería de pruebas:

Caso	Entrada	Salida esperada	Salida obtenida
1	...	...	
...	...	...	

Slide 4

Enunciado: Resolver la ecuación de segundo grado $ax^2 + bx + c = 0$ dados tres números reales como coeficientes.

Entradas: a, b, c, números reales, coeficientes de la ecuación
Salidas: Las soluciones de $ax^2 + bx + c = 0$.
Si son reales, sus valores;
si es una real, su valor, indicando que es doble;
si son complejas conjugadas, la parte real y la imaginaria
Objetivo: Resolver la ecuación de segundo grado
Método: $(-b \pm \sqrt{b^2 - 4ac})/(2a)$
Suposiciones: $a \neq 0$ Si $a = 0$, se responderá con un mensaje
Entorno local: Variables: x_1, x_2 para las soluciones reales
Re e Im para partes real e imaginaria
$Disc$ para el discriminante
Usa: Función de cálculo de $\sqrt{}$

Slide 5

Batería de pruebas:

Caso	Entrada	Salida esperada	Salida obtenida
1	1, 0, -9	3, -3	
2	2, 8'4, 8'42	-2'1, doble	
3	1, 1, 1	$-0'5 \pm 0'8660254i$	
4	0, 3, 5	'No es de segundo grado'	

Slide 6

Entradas: a, b, c, números reales, coeficientes de la ecuación
Salidas: Las soluciones de $ax^2 + bx + c = 0$
Si son complejas, su parte real e imaginaria
Objetivo: Resolver la ecuación $ax^2 + bx + c = 0$
Método: $(-b \pm \sqrt{b^2 - 4ac})/(2a)$ si $a \neq 0$
$-c/b$ si $a = 0$ y $b \neq 0$
'No hay solución' si $a = b = 0$ y $c \neq 0$
'Cualquier número es solución' si $a = b = c = 0$
Suposiciones:
Entorno local:Variables: x_1, x_2 para las soluciones reales
Re e Im para las partes real e imaginaria
$Disc$ para el discriminante
Usa: Función de cálculo de $\sqrt{}$

Slide 7

Batería de pruebas:

Caso	Entrada	Salida esperada	S. obtenida
1	1, 0, -9	3, -3	
2	2, 8'4, 8'42	-2'1, doble	
3	1, 1, 1	$-0'5 \pm 0'8660254i$	
4	0, 3, 5	-1'6666666	
5	0, 0, 1	'No hay solución'	
6	0, 0, 0	'Cualquier número es solución'	

Slide 8

Enunciado: Dados dos números enteros estrictamente positivos, obtener su máximo común divisor.

Entradas: m, n números enteros estrictamente positivos
Salidas: Su MCD
Método: Algoritmo de Euclides
Se basa en que, si $m \geq n$ y r es el resto de m/n:
(1) Si $r = 0$, n divide a m y el MCD es n
(2) Si no, $MCD(m, n) = MCD(n, r)$, y se rebaja el problema a un par de números menores.
Repitiendo el proceso, se llegará a la situación (1)
Suposiciones: $m, n > 0$ Si no, se responderá con un mensaje
Entorno local:
Variables: r para el resto de la división

Slide 9

Batería de pruebas:

Caso	Entrada	Salida esperada	Salida obtenida
1	120, 36	12	
2	225, 49	1	
3	225, 5	5	
4	0, 7	'No tiene sentido'	

Slide 10

Enunciado: Expresar en binario un número entero estrictamente positivo.

Entradas: n número entero estrictamente positivo (en decimal)
Salidas: La representación de n en binario
Método: Dado el número mayor posible de la forma 2^p
que se pueda restar de n,
escribir el bit correspondiente, y restarlo de n
rebajar 2^p,
hasta conseguir el último bit.
Suposiciones: $n > 0$ Si no, se responderá con un mensaje
Entorno local:
Variables: *maxpot* para el número de la forma 2^p

Slide 11

Batería de pruebas:

Caso	Entrada	Salida esperada	Salida obtenida
1	343	101010111	
2	8	1000	
3	-1	'No previsto'	
4	0	'No previsto'	

Slide 12

Diseño

- Diseño descendente
  - Refinamientos sucesivos ("*Stepwise Refinement*", Wirth)
  - Niveles de procesador
- Diseño modular
  - Divide y vencerás ("*Discurso del método*", Descartes)

Slide 13

Enunciado: Resolver la ecuación de segundo grado $ax^2 + bx + c = 0$ dados tres números reales como coeficientes.

Primer nivel de diseño Listado "Ecuación de segundo grado, primer nivel"

Segundo nivel de diseño

Listado "Ecuación de segundo grado, segundo nivel"

Slide 14

Enunciado: Dados dos números enteros estrictamente positivos, obtener su máximo común divisor.

Listado “MCD (máximo común divisor), primer nivel”

Slide 15

Nueva batería de pruebas:

Caso	Entrada	Salida esperada	Salida obtenida
1	36, 120	12	
2	225, 49	1	
3	225, 5	5	
4	0, 7	'No tiene sentido'	

Slide 16

Enunciado: Expresar en binario un número entero estrictamente positivo.

Listado "Expresar en binario un entero positivo"

Slide 17

Traza

- Prueba manual
- Simulación del comportamiento de un algoritmo para una instancia
- Refleja la evolución del *estado* del entorno:
  - entrada (**input**) por leer
  - salida (**output**) emitida
  - variables (sus valores)
  - punto de ejecución

Slide 18

Traza de MCD para 36 120

Listado: "MCD, segundo nivel"

Figura: "Traza para MCD 36 120"

Slide 19

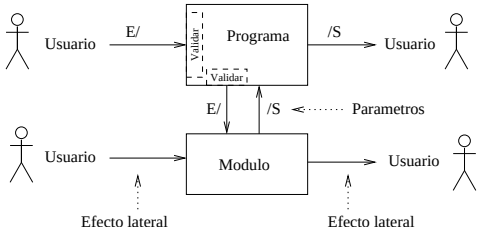
Módulos

- tarea determinada (acción o cálculo no elemental)
- análisis, diseño, codificación y pruebas independientes
- reutilizables
- elevan el nivel del lenguaje
- fragmentación y distribución del trabajo
- pueden definir subtareas (submódulos . . .)

Slide 20

Módulos

- tienen un nombre
- se comunican vía *parámetros*. Otra comunicación: "*efecto lateral*"



- son responsables de *cumplir la especificación*:

  si los argumentos de entrada cumplen la precondición
  entonces los de salida cumplirán la postcondición

- tipo *función* (cálculo) o *procedimiento* (acción)

Slide 21

Módulos. Pistas indicadoras de calidad

- nombre de elección casi evidente. Sintagma nominal para funciones, verbo para procedimientos. Sin conjunciones.
- pseudocódigo breve ($\simeq$ una página máximo), pero no demasiado breve (mínimo $\simeq$ 4 líneas).
- número de parámetros moderado.
- tarea dependiente únicamente de los parámetros (eventualmente puede no haberlos)
- efectos laterales: los mínimos posibles
- en general: máxima *cohesión* (fuerzas que unen el interior del módulo) y mínimo *acoplamiento* (dependencia entre módulos).

Slide 22

Verificación y validación

Verificación : demostrar la corrección:

"*el programa/módulo cumple la especificación*"

- Tiene éxito cuando lo demuestra
- Se basa en una formulación lógica del programa/módulo

Validación : incrementar la *fiabilidad* del programa/módulo

- Se basa en pruebas (de escritorio, trazas, en ejecución)
- Tiene éxito cuando encuentra un error
- Batería de pruebas "económica": que encuentre el mayor número de errores posible con el menor esfuerzo.

Depuración : estudio de las causas y consecuencias del error, localización, reparación (codificación, diseño, análisis) y actualización de la documentación

Slide 23

Ejemplo de especificación de módulo

Mayor potencia de 2 que se puede restar de un entero

Módulo: ILog2
Tipo: Función
Entrada (parámetro): m>0, entero
Valor Devuelto: la mayor potencia de 2 que sea ≤ m
Pre: m>0
Post: VD es potencia de 2, VD ≤ m < 2*VD
Entorno: Variables: maxpot: entero
Batería de pruebas: ...
Algoritmo: Listado "ILog2"

Slide 24

Ejemplo de especificación de módulo

División euclídea

Módulo DivMod
Tipo: Procedimiento
Entrada (parámetros): m, n : enteros, estrictamente positivos
Salida (parámetros): q, r : enteros positivos
Objetivo: Obtener cociente y resto de la división entera
Pre: $m \geq 0, n > 0$
Post: $m = nq + r \ \wedge \ 0 \leq r < n$

Slide 25

Ejemplo de especificación de módulo

Intercambiar valores

Módulo Intercambia
Tipo: Procedimiento
Entrada (parámetros): m, n : tipo T
Salida (parámetros): m, n (los de entrada)
Objetivo: Intercambiar los valores de m y n
Pre: $m = m\_0, n = n\_0$
Post: $m = n\_0, n = m\_0$

Slide 26

Validación de datos de entrada
Sin respuesta ante dato inválido

Enunciado ...
Entrada: datos x, de tipo `t`, cumpliendo $p(x)$
Salida: ...
Objetivo: ...
Observaciones: si la entrada x no cumple $p(x)$, no se responde
Algoritmo:

```
Inicio
    leer datos
    si datos válidos entonces
        procesar datos
    fin si
Fin
```

Slide 27

Validación de datos de entrada
Mensaje de datos inválidos

Enunciado ...
Entrada: datos x, de tipo `t`, cumpliendo $p(x)$
...
Observaciones: si no $p(x)$, se responde con un mensaje indicativo.
Algoritmo:

```
Inicio
    leer datos
    si datos válidos entonces
        procesar datos
    si no
        escribir mensaje de datos inválidos
    fin si
Fin
```

Slide 28

Validación de datos de entrada
Obtención de datos válidos

Módulo LeerDatosVálidos
Tipo: Procedimiento
Entrada: -
Salida: datos y válidos
Objetivo: conseguir del usuario datos válidos
Efecto lateral: lee del usuario
Algoritmo:

```
Inicio
    repetir
        leer datos
    hasta que datos válidos
Fin
```

Slide 29

Obtención de datos válidos con mensajes

Módulo LeerDatosVálidosInf
Tipo: Procedimiento
Entrada: -
Salida: datos válidos
Objetivo: obtener del usuario datos válidos; informa si no lo son.
Efecto lateral: lee del usuario y escribe "en pantalla"
Algoritmo:

```
Inicio
    leer datos
    mientras datos inválidos hacer
            escribir mensaje de error
            leer datos
    fin mientras
Fin
```