

Universidad Nacional de Jujuy

Facultad de ingeniería

Introducción a la informática

Clase 08

Modularización

UNJu-FI-Introducción a la Informática

Samuel Franco-José Zapana

1

Contenido

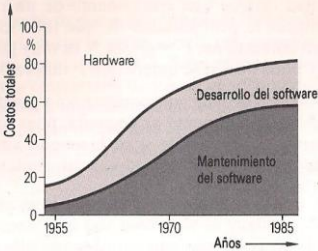
- Crisis del Software
- Modularización: concepto y justificación
- Diseño Top-Down
- Comunicación entre módulos
- Alcance de variables
- Efectos laterales
- Procedimientos y Funciones

UNJu-FI-Introducción a la informática

Samuel Franco-José Zapana

2

Desarrollo del Software



Distribución de los costos totales en hardware y software

UNJu-FI-Introducción a la Informática

Samuel Franco-José Zapana

3

Crisis del Software

- Término utilizado en 1968, en la primera conferencia organizada por la OTAN sobre desarrollo de software,
- Refiere a la dificultad de escribir programas libres de defectos (bugs), fácilmente comprensibles y que sean verificables (fines de la década del 60)

UNJu-FI-Introducción a la informática

Samuel Franco-José Zapana

4

Crisis del Software

- Causas:
 - Complejidad de la forma de programar
 - Cambios a los que se ven afectados los programas
 - Ausencia de herramientas para estimar el esfuerzo
- Sucesos que se observan
 - Los proyectos no terminan en plazo
 - Los presupuestos no se ajustan al presupuesto inicial
 - Baja calidad del software generado
 - Software que no cumplía con las especificaciones
 - Código difícil de mantener

UNJu-FI-Introducción a la Informática

Samuel Franco-José Zapana

5

Factores de influencia

Para poder llevar el estado del proceso de software como un estado de crisis, los críticos han destacado ciertas características que han permitido esta postura del software respecto a otras etapas de su corta historia. Algunos de esos factores son:

- Aumento del poder computacional.
- Reducción del costo del hardware.
- Rápida obsolescencia de hardware y software.

UNJu-FI-Introducción a la informática

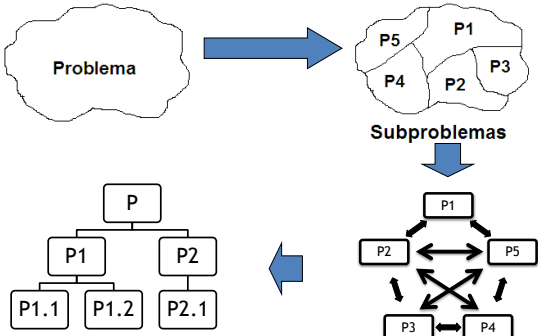
Samuel Franco-José Zapana

6

Estrategia: Divide y conquista

- Es fácil resolver pequeños problemas pero es difícil resolver los grandes problemas.
- Los diseñadores resuelven un problema separando un algoritmo en partes que son más manejables.
- Las partes son resueltas individualmente; luego las soluciones pequeñas son integradas y se obtiene la gran solución.
- Este proceso se denomina descomposición, “divide y conquista” o mas comunmente diseño...*top-down*.

Complejidad

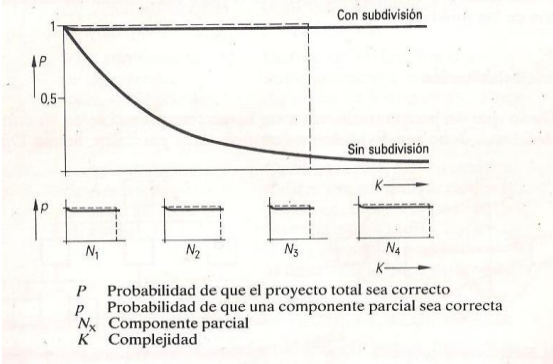


Modularización: Justificación



“Si la probabilidad de que una componente de un programa sea correcta es p , entonces la probabilidad de que todo el programa de N componentes sea correcto es $P = p^N$.
Si N es muy grande, entonces p debe diferir muy poco de uno si se quiere que P difiera de cero”
Edsger Dijkstra

Modularización: Justificación



Modularización

- Si un problema es difícil de modularizar, hay que intentar redefinir las tareas involucradas en el problema.
- Demasiados casos especiales o demasiadas variables que requieren un tratamiento especial, son signos de que la definición del problema es inadecuado

En consecuencia los módulos

- Deberán estar jerarquizados
- Deberán ser pequeños, sencillos e independientes
- Deberán ser legibles
- Deberán usar una estructura de datos y control adecuados

Modularización

Ventajas

- Simplifica el Diseño.
- Disminuye la complejidad de los algoritmos.
- Disminuye el tamaño total del programa.
- Promueve la reusabilidad.
- Promueve la abstracción
 - Características de caja negra.
 - Aislamiento de los detalles mediante.
- Favorece el trabajo en equipo.
- Facilita la depuración y prueba.
- Facilita el mantenimiento.
- Permite la estructuración de librerías específicas.

Modularización

Desventajas

- La integración de los módulos puede ser compleja en particular si fueron escritos por diferentes personas.
- Los módulos requieren de documentación cuidadosa dado que pueden afectar a otras partes del programa. Se debe prestar mucha atención cuando se utilizan módulos que modifican estructuras de datos compartidas con otros módulos.
- La prueba de un modulo puede ser difícil en situaciones en las que se necesitan datos generados por otros módulos.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

13

Llamado a un módulo

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

14

Definición de módulo

Un **módulo** es una rutina, subrutina, subalgoritmo, **procedimiento** o **función** que puede definirse dentro de un algoritmo u otro módulo superior con el fin de ejecutar una tarea específica y puede ser llamado o invocado desde el algoritmo principal o desde un módulo superior cuando sea necesario.

Función

Procedimiento

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

15

Diseño Top-Down: Comunicación

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

16

Términos

- A las variables definidas en el contexto general, se les llama **variables globales** y pueden ser utilizadas en cualquier parte del algoritmo.
- Las **variables locales** definidas en un determinado módulo, sólo pueden utilizarse en el contexto de dicho módulo.
- La **lista de argumentos** es un medio de comunicación entre los distintos módulos en forma bidireccional no necesariamente.
- Los **parámetros** son variables o valores que son copiados en los argumentos de un módulo, para que éste realice su tarea específica.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

17

Comunicación: Llamado y recepción

Las subrutinas parametrizadas toman argumentos que controlan aspectos de su comportamiento o especifican los datos sobre los que operan.

Tipos de parámetros

- Parámetros Formales:** aparecen en la declaración de la subrutina.
- Parámetros Actuales:** variables y expresiones que son pasadas a la subrutina en el momento de la invocación.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

18

Modo de Comunicación

Clasificación de parámetros por su forma de envío

- **Por Valor:** cada parámetro actual es asignado en el parámetro formal correspondiente cuando una subrutina es llamada. Debe ser tratado como una variable de la cuál **el módulo hace una copia** y la utiliza localmente.
- **Por Referencia:** cada parámetro formal es un alias del parámetro actual correspondiente, significa que el módulo recibe la dirección de memoria de una variable conocida en el punto de invocación. **Operan directamente sobre la dirección de la variable original**, en el contexto del módulo que llama, esto significa que no requiere memoria local.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

19

Modo de Comunicación

El número, orden y tipo de los parámetros o argumentos utilizados en la invocación a una Función o un Procedimiento **deben coincidir con el número, orden y tipo de parámetros** del encabezamiento del módulo.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

20

Alcance de las Variables

- **Variable local:** Es aquella **declarada dentro de un subprograma** y es distinta de las variables que tengan el mismo nombre en cualquier parte del programa principal. Cuando se hace referencia a una variable con el mismo nombre que otra dentro de un programa, se refiere a una posición diferente de memoria.
- **Variables Globales:** Es aquella que **está declarada en el programa principal** y se puede referenciar desde cualquier parte del programa, inclusive desde otros subprogramas.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

21

Ámbito de las variables

```
graph TD; P["P  
A, B, C"] --> P1["P1  
A, D"]; P --> P2["P2  
C, D, F"]; P1 --> P1_1["P1.1  
E, C"]; P1 --> P1_2["P1.2  
A"]; P2 --> P2_1["P2.1  
D"];
```

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

22

Comunicación ≠ Efectos laterales

- La comunicación entre el programa principal (o módulos) con los demás módulos, **debe realizarse a través de parámetros**. Cualquier otra comunicación se conoce como efectos laterales.

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

23

Funciones: Declaración

- Representan rutinas de programas que son invocadas desde algún otro programa.
- Son referenciadas a través de un nombre en una expresión y una lista de parámetros actuales y **devuelve un valor**.

funcion <Id> (<parámetros><tipo><modo>): <tipo>
[declaraciones de variables locales]
inicio
<acciones> /*cuerpo de la función*/
devolver (<Id>)
finfuncion

UNJu-FI-Introducción a la Informática
Samuel Franco-José Zapana

24

Procedimientos: Declaración

- Es un subprograma que ejecuta un proceso específico.
- Ningún valor está asociado con el nombre del procedimiento.
- A diferencia de las funciones no devuelven valores.

```
procedimiento <Id> (<parámetros><tipo><modo>)  
[declaraciones locales]  
inicio  
  <acciones> /*cuerpo del procedimiento*/  
finprocedimiento
```

Procedimientos vs Funciones

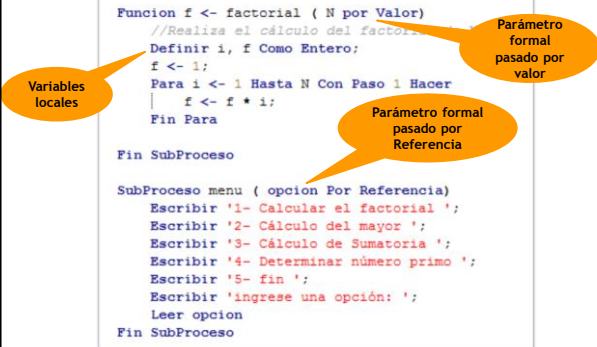
- Un procedimiento es llamado desde el proceso principal o un módulo superior mediante su nombre y una lista de parámetros actuales.
- Las funciones devuelven un valor
- Los procedimientos pueden devolver 0, 1 o N valores y en forma de lista de parámetros.
- El procedimiento se declara igual que la función pero su nombre NO esta asociado a ninguno de los resultados que obtiene.

Algunas formas de llamar a una Función

- En una asignación:
p <- factorial(5)
- En una operación escribir:
Escribir factorial(5)
- En una expresión algebraica
y <- 1 / factorial(x) - 1 / factorial(z)
- En una condición
Si (y > factorial(x)) Entonces ...
Si (primo(x)) Entonces ...
- Otras...

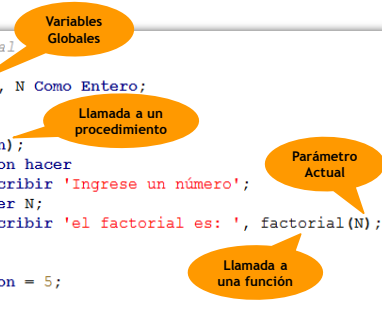
Ejemplo

```
Funcion f <- factorial ( N por Valor)  
//Realiza el cálculo del factorial  
Definir i, f Como Entero;  
f <- 1;  
Para i <- 1 Hasta N Con Paso 1 Hacer  
| f <- f * i;  
Fin Para  
  
Fin SubProceso  
  
SubProceso menu ( opcion Por Referencia)  
Escribir '1- Calcular el factorial '  
Escribir '2- Cálculo del mayor '  
Escribir '3- Cálculo de Sumatoria '  
Escribir '4- Determinar número primo '  
Escribir '5- fin '  
Escribir 'ingrese una opción: '  
Leer opcion  
Fin SubProceso
```



Ejemplo

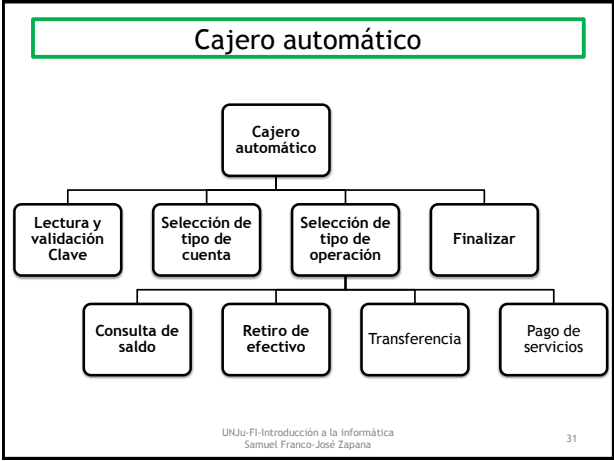
```
//Programa principal  
Proceso Ejemplo02  
Definir opcion, N Como Entero;  
saludo();  
Repetir  
| menu(opcion);  
| Segun opcion hacer  
| | 1 : Escribir 'Ingrese un número';  
| | Leer N;  
| | Escribir 'el factorial es: ', factorial(N);  
| FinSegun  
Hasta Que opcion = 5;  
FinProceso
```



Serie de Taylor del coseno x

- Función Potencia
- Función División
- Función Factorial

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad \text{para todo } x$$



Bibliografía

- Fundamentos de Programación: Joyanes Aguilar - Libro de problemas (Capítulo 5)

Recuerde que le estudio de estos temas debe ser complementado con la lectura de un libro

UNJu-FI-Introducción a la informática
Samuel Franco-José Zapana

32