Clase 6

Formas de comunicarse entre rutinas

Rutinas

Conjunto de sentencias que representan acción abstracta. Representan una unidad de programa. Se dice que permite definir una operación creada por el usuario a semejanza de las operaciones primarias integradas en el lenguaje. Ayuda a modularizar, agrega algo que el lenguaje no tiene. Puede que devuelvan algo o no.

Formas de subprogramas:

- Procedimientos
    - Permite dar nombre a un conjunto de sentencias y declaraciones asociadas
    - Proveen solución más corta, comprensible y fácilmente modificable.
    - Los resultados los produce en variables no locales o en parámetros que cambian su valor.
    - Pueden no recibir ni devolver ningún valor.
    - Es el programador quien define y crear nuevas acciones agrupando sentencias.
- Funciones
    - Devuelve SIEMPRE un valor al punto donde se llamó
    - Permite al programador crear nuevas operaciones
    - Similar a las funciones matemáticas, ya que solo devuelven un valor.
    - Se las invoca dentro de expresiones y el valor que devuelve remplaza a la invocación dentro de la expresión.

Formas de compartir datos entre diferentes unidades:

Si las unidades utilizan variables locales no hay problema

Si las variables no son locales hay 2 formas:

1) A través de un ambiente no local

    Ambiente común explícito:
    - Se definen áreas comunes de código
    - El programador debe especificar que comparte
    - Cada lenguaje tiene su forma de realizarlo

    Ambiente no local implícito
    - Es automático
    - Utilizando regla de alcance dinámico
    - Utilizando regla de alcance estático

2) Pasaje de Parámetros

Parámetro Real (Argumento): Es un valor u otra entidad utilizada para pasar a un procedimiento o función. Están en la parte de la invocación

Parámetro Formal (Parámetro): es una variable utilizada para recibir valores de entrada en una rutina, subrutina etc. Se ponen en la parte de la declaración. Es una variable local a su entorno.

Ventajas uso de pasaje de parámetros:

- Con el uso de pasaje de parámetros tengo más control.
- Enviar distintos parámetros en distintas invocaciones
- Flexibilidad
- Legibilidad
- Protección
- Abstracción

- Evaluación: en general, cuando se invoca, se evalúa los parámetros reales, y luego se hace la ligadura antes de transferir el control a la unidad llamada

- Ligadura:
    - Posicional: Se corresponden con la posición que ocupan en la lista de parámetros. Van en el mismo orden
    - Palabra clave o nombre: Se corresponden con el nombre por lo tanto pueden estar colocados en distinto orden en la lista de parámetros.

Tipos de parámetros:

- Datos
- Subprogramas

💡 Las funciones no deberian producir efectos laterales. Es decir, no deben alterar el valor de ningun dato, solo producir resultados. Los parametro formales deberian ser siempre modo IN como ada. Por otro lado deberia ser ortogonal, eso quiere decir, que los resultados deberian poder ser de cualquier tipo como ada y no como pascal por ejemplo, que solo permite escalares.

1. Datos como Parámetros

Desde el punto de vista semántico de los parámetros formales pueden ser:

Modo IN (mecanismo default):

- El parámetro formal recibe el dato desde el parámetro real. (la conexión es al inicio y se corta la vinculación)
- Posee dos tipos:
    - Por valor: Se transfiere el dato real y se copia. El parámetro formal actúa como una variable local de la unidad llamada, y crea otra variable.
        - Ventajas: El parámetro real se protege, ya que se hace una copia, por lo que el "verdadero" no se modifica. No hay efectos colaterales.

        - Desventaja: Consume tiempo para hacer la copia y ocupa más almacenamiento ya que el dato se duplica.
    - Por valor contante: es lo mismo que el de valor, pero tiene la particularidad de que no se puede modificar su valor. Ejemplo: parámetros In en ADA.
        - Ventajas: Protege los datos de la unidad llamadora, el parámetro real no se modifica.
        - Desventajas: Requiere realizas más trabajo para implementar los controles.

Modo OUT:

- El valor del parámetro formal se copia al parámetro real al terminar de ejecutarse la unidad que fue llamada (rutina)
- Posee dos tipos:
    - Por resultado: El parámetro formal es una variable local. El parámetro formal es una variable sin valor inicial porque no recibe nada. Debemos inicializar de alguna forma si el lenguaje no lo hace por defecto. Una vez que se vuelve asigna el valor a devolver (sobreescribo la variable que pase como parámetro al terminarse la rutina)
        - Ventajas: protege los datos de la unidad llamadora, el parámetro real no se modifica en la ejecución de la unidad llamada
        - Desventajas: Consume tiempo y espacio porque hace copia al final. Debemos inicializar la variable en la unidad llamada de alguna forma (si el lenguaje no lo hace por defecto)
    - Por Resultado de funciones: Es el resultado que me devuelven las funciones. Remplaza la invocación en la expresión que contiene el llamado, es decir, debo hacer una asignación para que ese reemplazo tenga sentido.

Modo IN/OUT:

- El parámetro formal recibe el dato del parámetro real y el parámetro formal le envía el dato al parámetro real al finalizar la rutina. (la conexión es al inicio y al final)
- La conexión no es todo el tiempo: evalua, conecta ligadura, ejecuta, conecta ligadura.
- Posee tres tipos:
    - Por Valor/Resultado: El parametro formal es una variable local que recibe una copia (a la entrada) del contenido del parametro real y el parametro real (a la salida) recibe una copa de lo que tiene el parametro formal. Basciamente Lo que hace la rutina lo copio en la variable. Cada referencia al parametro formal es una referencia local.
        - Tienen las ventajas y desventaja de cada uno.
    - Por Referencia: Comparte la direccion del parametro real al parametro formal. Como si fuese una especie de puntero el parametro formal. El parametro formal va a ser una variable local a la unidad llamadora que contiene la direccion en el ambiente no local. Cualquier cambio que se realice en el parametro formal dentro del cuerpo del subprograma quedara registrado en el parametro real.
        - Ventajas: eficiente en tiempo y espacio ya que no se realizan copias de datos.

        - Desventajas: El acceso a datos es lento por la indireccion. Es posible que se modifque el dato sin querer. Es muy dificil la verificacion de programa, esto se debe a que se pueden generar alias que afecten la legibilidad.
    - Por nombre: El parámetro formal es sustituido textualmente por una expresión del parámetro real más un puntero al entorno del parámetro real. (se maneja una estructura aparte que resuelve esto). Se establece la ligadura entre parámetro formal y parámetro real en el momento de la invocación, pero la "ligadura de valor" se difiere hasta el momento en que se lo utiliza (la dirección se resuelve en ejecución). Distinto a por referencia. Es decir, no apunta a una dirección fija, puede ir cambiando (pero el nombre tiene que si ser el mismo)
        - Ventajas: Es un metodo mas flexible pero mas lento, ya que debe evaluarse cada vez que se usa.
        - Desventajas: Es dificil de implementar y genera soluciones confusas para el lector y el escritor.

Pasaje de parametros en algunos lenguajes:

C

- Por valor (Si se necesita por referencia se usa punteros)
- Permite pasaje por valor constante, agregando const

Pascal

- Por valor (por defecto)
- Por referencia (opcional : var)

C++

- Igual que C más pasaje por referencia

Java

- El único mecanismo contemplado es el paso por copia de valor. Pero como las variables de tipo no primitivos son todas referencias a variables anónimas en la heap, el paso por valor de una de estas variables son en realidad un paso por referencia de las variables

Python

- Se puede pasar de dos formas:
    - Inmutables: actuara como por valor
    - Mutables: No se hace una copia sino que se trabaja sobre el.

PHP

- Por valor de forma predeterminada Utilizando "&" por referencia

Ruby

- Por valor, pero al igual que python, si se pasa un objeto "mutable", no se hace una copia sino que se trabaja sobre el.

ADA

- Por defecto con copia IN
- Por resultado OUT
- IN OUT
- Para los tipos primitivos indica por valor-resultado
- Para los tipos no primitivos, datos compuestos (arreglo , registro) se hace por referencia.

Caso particular de ADA: En las funciones solo se permite el paso por copia de valor, lo cual evita la posibilidad de efectos laterales.

2. Subprogramas como parámetros.

- No lo tienen todos los lenguajes.
- Debe determinarse cuál es el ambiente de referencia no local correcto para un subprograma que se ha invocado y que ha sido pasado como parámetro.
- Hay varias opciones:
    - Ligadura shallow o superficial: el ambiente de referencia es el del subprograma que tiene el parámetro formal para el subprograma que se está pasando como parámetro real.
    - Ligadura deep o profunda: el ambiente de referencia es el del subprograma donde está declarado el subprograma usado como parámetro real. Este enfoque se utiliza en lenguajes con alcance estático y estructura de bloque.
    - El ambiente del subprograma donde se encuentra el llamado a la unidad que tiene un parámetro subprograma. Poco natural.
- Cuando los lenguajes no tienen para enviar subprogramas como parámetros utilizan unidades genéricas