Clase 6-Rutinas y Parametros

Formas de comunicación entre las rutinas – Parametros:

Rutinas: estan formadas por un conjunto de sentencias que representan una accion abstracta, permiten al programador definir una nueva operación y de esta forma ampliar a los lenguajes, dan modularidad, claridad y buen diseño. Se lanzan con una llamada explicita (se invocan por su nombre) y luego retornan a algun punto de la ejecucion (responden al esquema call/return). Los subprogramas son el mejor ejemplo.

Formas de subprogramas:

.Procedimientos: el programador las invocara, pueden no recibir ni devolver ningun valor. Ademas, los resultados los produce en variables no locales o en parametros que cambian su valor.

.Funciones: una funcion devuelve un valor al punto donde se llamo. El valor que recibe la funcion se usa para calcular el valor total de la expresion y devolver algun valor. Se las invoca dentro de expresiones y lo que calcula remplaza a la invocacion dentro de la expresion. Siempre deben retornar un valor.

.Un subprograma es la implementacion de una accion abstracta y su invocacion representa su uso.

Formas de conectar y “compartir datos” entre diferentes unidades de programa:

1.Variables locales (no hay problema)

2.Variables no locales:

A.Acceso al ambiente no local (puede llevar a errores, puede ser menos claro)

.Ambiente no local implicito (es automatico)

.Utilizando regla de alcance dinamico

.Utilizando regla de alcance estatico

.Ambiente comun explicito (interviene el programador)

.Se definen areas comunes de codigo, el programdor especifica que comparte y cada lenguaje tiene su forma de realizarlo

B.A traves del uso de parametros (la mejor forma de compartir y mas clara)

Pasaje de parametros:

.Parametro real (argumento): es un valor u otra entidad que se pasa a un procedimiento o funcion. Estan colocados en la parte de la invocacion de la rutina

.Parametro formal (parametro): es una variable utilizada para recibir valores de entrada en una rutina, subrutina, etc. Estan colocados en la parte de la declaracion de la rutina

El pasaje de parametros es mejor porque el uso intensivo de accesos al ambiente no local puede provocar alguna perdida de control, y puede provocar que las variables terminen siendo visibles donde no es necesario llevar a errores. Ademas entre sus ventajas estan la flexibilidad, la abstraccion, la proteccion, la legibilidad y la modificabilidad que nos otorga su uso.

Parametros: Los parametros formales se consideran una variable local a su entorno. Un parametro real puede ser un valor, entidad, etc. Que pueden ser locales, no locales o globales y que se especifica en la llamada a una funcion/rutina dentro de la unidad llamante.

.El momento de vinculacion entre los PR y PF comprende la evaluacion de los PR y la ligadura con los PF.

.Evaluacion: en general es antes de la invocacion, primero se evaluan los parametro reales, y luego se hace la ligadura. Se verifica que todo este bien antes de transferir el control a la unidad llamada.

.Ligadura: puede ser de dos tipos (varian según cada lenguaje y cada uno tiene sus reglas sintacticas y semanticasa para el pasaje):

.Por posicion: se corresponde con la posicion que ocupan en la lista de parametros. Van en el mismo orden.

.Por nombre o palabra clave: se corresponden con el nombre por lo tanto pueden estar colocados en distinto orden en la lista de parametros.

Tipos de parametros (cada uno tiene distintos subtipos y depende de cada lenguaje):

1.Datos pasados como parametros: hay diferentes formas de transmitir los parametros hacia y desde la rutina llamada. Desde el punto de vista semantico la clasificacion del tipo de comunicación permitida para los parametros puede ser de tres tipos y la forma de transmitir los paramteros puede ser de dos maneras:

A.IN: el parametro formal recibe el dato desde el parametro real. La conexión es al inicio, se copia y se corta la vinculacion.

.IN-Por valor: el valor del PR se usa para inicializar el PF al invocar la unidad. Se transfiere el dato real y se copia, en este caso el PF actua como una var local de la unidad llamada, y crea otra variable. Esto consume tiempo para hacer la copia y almacenamiento pero se protegen los datos.

.IN-Por valor constante: no es permitido por todos los lenguajes. Se envia un valor, pero la unidad receptora no puede modificarlo, queda con un valor fijo. La implementacion debe contemplar que el parametro real no sea modificado. Esto lleva mas trabajo para realizar los controles pero se protegen los datos.

B.OUT: se invoca la rutina y cuando esta termina devuelve el parametro formal al parametro real. La conexión es al final.

.OUT-Por resultado: el valor del PF se copia al PR al terminar de ejecutarse, el PF es una var local del entorno de la rutina, el PF es una varible sin valor inicial porque no recibe nada (puede que tenga una regla de inicializacion por defecto del lenguaje). Consume tiempo y espacio al hacer la copia, ademas se debe inicializar la variable en la unidad llamada pero protege los datos.

.OUT-Por resultado de funciones: resultado que devuelven las funciones, el resultado se remplaza en la invocacion de la expresion que contiene el llamado. La forma de la devolucion depende del lenguaje. Return (valor o expresion), Nombre de la funcion (ultimo valor asignado) que se considera como una variable local.

C.IN/OUT: el parametro formal recibe el dato del parametro real y el parametro formal le envia el dato al parametro real al finalizar la rutina. La conexión es al inicio y al final.

.IN-OUT-Por valor/resultado: la rutina recibe un valor y devuelve un resultado. Cuando se invoca la rutina el PR le da valor al PF (se genera copia) y se desliga en ese momento, mientras que la rutina trabaja las referencias al PF son referencias locales, una vez que termina de ejecutar el PF le devuelve un valor al PR y se genera copia. Se dice que hay una ligadura y una conexión entre PR y PF cuando se inicia la ejecucion y cuando se termina, pero NO en el medio.(ventajas y desventajas de ambos)

.IN-OUT-Por referencia: tambien llamada por variable, no es copia por valor sino que es copia por referencia a una posicion, se asocia la direccion (l-valor) del PR al PF. La conexión es la inicio y PERMANECE hasta el final. El PF sera una variable local a su entorno pero que contiene la direccion al PR de la unidad llamadora que estara entonces en un ambiente no local. Asi se extiende el alcance de la rutina (aliasing situation). Cada referencia al PF sera a un ambiente no local, entonces cualquier cambio que se realice en el PF dentro del subprograma quedara registrado en el PR. El cambio sera automatico. El PR queda compartido por la unidad llamadora y llamada. Sera bidireccional.

.IN-OUT-Por nombre: introducido por ALGOL60. No es utilizado hoy en ningun lenguaje imperativo. El PF es sustituido “por una expresion textual” del PR, mas un “puntero al entorno” del PR. Se utiliza una estructura aparte que resuelve esto. La ligadura de parametros(entre PF y PR) es en el momento de la invocacion. La “ligadura de valor” se difiere hasta el momento en que se lo utiliza (la direccion se resuelve en ejecucion). Su objetivo es otorgar evolucion de valor diferida.

.Thunks: es una unidad pequeña de codigo (funcion) que encapsula y representa a una expresion que pospone su evaluacion hasta que sea necesario. Es un concepto clave en la “evaluacion perezosa/diferida”

.Para implementar el pasaje por nombre se utilizan: los thunks, cada aparicion del PF se reemplaza en el cuerpo de la unidad llamadapor una invocacion a un thunks, en el momento de ejecucion activara al procedimiento que evaluara el PR en el ambiente apropiado.

.Pasaje de parametros en algunos lenguajes (pag 56,57,58)

2. Parametro subprograma:

A.Rutinas que reciben funciones como parametros.

Hay situaciones en que es conveniente o necesario poder usar nombres de subprogramas como parametro para ejecutar alguna accion. Con el uso de este tipo de parametros surgen dudas que hay que resolver tales como: chequeo de tipos de subprogramas, subprogramas anidados y ¿Cuál es el entorno de referencia?

.Ambiente de referencia para las referencias no locales dentro del cuerpo del subprograma pasado como parametro:

.Debe determinarse cual es el ambiente de referencia no local correcto para un subprograma que se ha invocado y que ha sido pasado como parametro. Para esto se usa la cadena estatica o cadena dinamica.

Existen tres condiciones “especiales”:

1.Se pasan funciones como argumentos.

2.Esas funciones tienen referencias a variables definidas fuera de ellas.

3.Hay funciones anidadas.

Para ello existen tres reglas de alcance y binding de parametros:

1.Ligadura shallow (superficial): mira la unidad donde esta declarado el subprograma que contiene a la funcion como PF (subprograma que lo invoca).

2.Ligadura deep (profunda): mira la unidad donde esta declarado el subprograma parametro.

3.Ligadura ad-hoc: el ambiente de referencia es el del subprograma donde se encuentra el llamado a la unidad que tiene un parametro de tipo subprograma.

B.Rutinas que devuelven funciones como resultado (mecanismo fundamental en lenguajes funcionales).