Clase 4

Unidades

- Es de lo que están compuestos los programas. Es una acción abstracta.

Generalmente de las llamas rutinas. Estas rutinas se pueden clasificar en

  - Procedimientos.
  - Funciones (Devuelven un valor).

Nota: No todos los programas implementan los dos. Hay lenguajes que SOLO tienen "funciones" y "simulan" los procedimientos con "funciones que devuelven void".

Atributos:

Nombre (Identificador): Es un String de caracteres que se usa para invocar (y declarar) a la rutina. Puedo llamarla dentro del alcance.

Alcance: Rango de instrucciones donde se conoce su nombre. El alcance se extiende desde el punto de su declaración hasta algún constructor de cierre. Según su lenguaje puede ser estático o dinámico.

- Hay que distinguir definición (el cuerpo) de la declaración (encabezado). Se puede invocar después de la declaración (puede ser declaración sola sin definición)

Tipo: El encabezado de la rutina define el tipo de los parámetros y el tipo del valor de retorno (si lo hay) ¿Para qué sirve? Para hacer chequeo de tipos. Debe haber una correspondencia entre los tipos de los parámetros formales y los reales.

L-valor: Es el lugar de memoria en que se almacena el cuerpo de la rutina (lo que hace)

R-valor: Dirección del código que se va a remplazar en el programa. La llamada a la rutina causa la ejecución su código, eso construye su r-valor.

- Puede ser estático (el caso más usual) o dinámico (se implementa a través depunteros a rutinas).

Comunicación entre rutinas

Ambiente no local: No es local, por lo que tiene que ir a buscarlo. (acotado, son las variables globales)

Parámetros: Mas legible.

Parámetros formales: Se especifican en la definición.

Parámetros reales: Aparecen en la invocación de la rutina.

Ligadura entre parámetros formales y reales

Relacionar parámetros formales y reales mediante

- Método posicional: Se ligan según la posición en la llamada y en la definición.
- Variante: Combinación de lo anterior con valores por defecto
- Método por nombre: Se ligan por el nombre. Se debe conocer los nombres de los parámetros formales.

Representación en ejecución

Cuando se invoca una rutina se ejecuta una instancia del proceso con los particulares valores de los parámetros.

Se le llama instancia de unidad a la representación de la rutina en ejecución. La cual posee dospartes:

- *Segmentos de código (Contenido fijo):* Instrucciones de la unidad se almacena en la memoria de instrucción C
- *Registro de activación (Contenido cambiante):* Datos locales de la unidad se almacena en la memoria de datos D

Procesador abstracto

Para comprender que efecto causa las instrucciones del lenguaje al serejecutadas. Se describe la semántica del lenguaje de programación a través de reglas de cada constructor del lenguaje traduciéndolo en una secuencia de instrucciones equivalentes del procesador abstracto.

- Memoria de código: C(y), donde y es una dirección. Dirección de la memoria de código,
- Memoria de datos: D(y), donde y es una dirección o una celda de memoria. Celda de memoria de datos
- IP: Puntero a la instrucción que se está ejecutando. Luego de que se ejecuta una instrucción se aumenta y se pasa a la siguiente.

Instrucciones

- SET: Asignación valores en la memoria de datos. Ej.: *set tercer, source.* → Copia el valor representado en *source* en la dirección representadapor *target*.
- E/S: *read* y *write* permiten la comunicación con el exterior
- JUMP: bifurcación incondicional. Ej.: *jump 47* → La próxima instrucción a ejecutarse será la que este almacenada en la dirección 47 de C.
- JUMPT: bifurcación condicional, bifurca si la expresión se evalúa como verdadera. Ej.: *jumpt 47, D [13] > D [8]*

Puntos de retorno: Información que se debe guardar cuando una unidad es llamada. Contiene la siguiente dirección a ejecutar después de que termine la subrutina

Ambiente de referencia

- Ambiente local: Variables locales, están almacenadas en su registro de activación.
- Ambiente no local: Variables no locales, están en los registros de activación de otras unidades.

Estructura de ejecución de los lenguajes de programación

Estático: espacio fijo.

- El espacio necesario para la ejecución se deduce del código
- Todos los requerimientos de memoria necesarios se conocen antes de la ejecución
- La alocación puede hacerse estáticamente. Las unidades en la memoria perduran durante toda la ejecución del programa.

- No puede haber recursión

Basado en pilas (pascal, java, c): espacio predecible.

- El espacio se deduce del código.
- La memoria a utilizarse es predecible y sigue una disciplina last-in-first-out.
- Las variables se alocan automáticamente y se desalocan cuando el alcance se termina

Dinámico (Python, Ruby): espacio impredecible.

- Todo se maneja de forma dinámica. Los datos se alocan en la zona de memoria heap.