Tipo de Dato Referencia

Se conoce como tipo de dato referencia, apuntador, o pointer. Una referencia no es más que un tipo de dato elemental que representa una dirección de memoria en donde por lo general se encuentra un dato, sea elemental o estructurado. Los apuntadores son la base para la creación de la mayoría de las estructuras dinámicas, como listas, árboles y grafos. La cantidad de memoria que ocupa cada variable de tipo referencia es 1 palabra. Se declara de la siguiente forma:

```
<tipo de dato> * <nombre-variable u objeto>
```

Algunos ejemplos:

```
Type Register Point
  Real x,y
end
Integer * pInt      // variable apuntador a entero
Point * pPoint      // variable apuntador al tipo Point
Type Point * PPoint // tipo apuntador a Point
PPoint pMyPointer   // variable apuntador a Point
```

Los valores asociados al tipo referencia son direcciones de memoria. Si se pueden direccionar 4 GB, entonces las direcciones van desde la posición de memoria 0 hasta $2^{32} - 1$ (haciendo abstracción de los mecanismos de direccionamiento). Esto va a depender del tamaño de la dirección en una arquitectura dada. La dirección 0 por lo general está reservada, y se utiliza para inicializar los apuntadores, o indicar que no están apuntando a ningún valor (constante NULL o NIL.

El tipo Refencia permite las operaciones de asignación y relacionales. Igualmente, otro operador es la dereferenciación. Este operador retorna el dato referenciado por el apuntador, y se coloca antes del identificador que se desea dereferenciar. Se empleará el símbolo * para dereferenciar y se acostumbra colocarlo entre paréntesis junto con el identificador para mayor legibilidad. Ejemplo:

```
*pPoint.x = 0 // se accede la coordenada x del punto pPoint, tambien se puede emplear como (*pPoint).x
Point ptNew   // se define una variable de tipo registro Point
ptNew = *pPoint // el punto apuntado por pPoint es asignado a ptNew
```

La operación opuesta a la dereferenciación es la referenciación. Este operador retorna la dirección de una variable u objeto. Se simboliza mediante la palabra **ref**

```
Point ptMyPoint;
Point * pPointer;
pPointer = ref ptMyPoint;   // pPointer toma la direccion en donde esta almacenada ptMyPoint
*pPointer.x = 4             // se asigna 4 al campo x de pPointer. Equivale a ptMyPoint.x = 4;
*(ref ptMyPoint).y = 3      // Equivalente a ptMyPoint.y = 3;
*(*(ref pPointer)).y = 1    // Equivalente a ptMyPoint.y = 1;
```

El operador * tiene prioridad sobre el operador punto . utilizado para acceder miembros de registros y objetos, y éste sobre el operador ref. Entre otros operadores sobre apuntadores se encuentra el new y delete.

Operador new: Esta función reserva el espacio en memoria suficiente para el tipo de dato especificado como parámetro y retorna la dirección de memoria que hace referencia a dicho espacio. Si no se logró reservar el espacio entonces retorna NULL o NIL. Es buena práctica de programación, verificar que se reservó la cantidad de memoria solicitada. En los compiladores, si no hay memoria, se genera una excepción.

Operador delete: Esta acción libera la memoria dinámica reservada mediante el operador new. Se debe colocar delete seguido del apuntador. Como efecto de la operación, o estado final, es que la memoria liberada ya está disponible para alojar nuevos datos, pero por estar libre, no puede ser accedida, pues generaría un fallo de protección de memoria.

```
// Ejemplo del uso del new y delete
Integer * pArray, pInt
pInt = new Integer
pArray = new Integer [3]
if pInt  == NIL or pArray == NIL then
  exit("Out of memory")
end
*pArray[1] = *pArray[2] = *pInt = 1
*pArray[3] = *(ref pArray + 1) + (*pInt) * 2
delete pInt
delete [] pArray
```

esmitt ramírez/2014