Cómo usar tipos de dato que son por referencia en lugar de por valor

Encontré una característica muy interesante acerca de los tipos de dato como **Listas** y **Diccionarios**.

Un tipo de dato **Lista** es definido como una lista de objetos que pueden ser accedidos por índice. Puedes chequear todos los métodos disponibles para el Tipo de dato Lista.

Un tipo de dato Diccionario es definido como una colección de llaves y valores. Puedes chequear todos los métodos disponibles para el Tipo de dato Diccionario.

La parte interesante es que por razones de rendimiento estos tipos de dato son por referencia y no por valor.

Tipo referencia significa que la variable no tiene un valor en si misma, sino que contiene una referencia a una instancia de un objeto en memoria. Cuando pasas una variable de este tipo como parámetro a una función, esa función está recibiendo una referencia al objeto en memoria y trabajará directamente sobre ese objeto a través de esa referencia.

Déjame explicar el concepto con un ejemplo simple:

Podemos extraer la lista de las facturas que están vencidas usando una tipo de dato Lista como lo muestra el siguiente código

Note que el parámetro no fue pasado como **var**

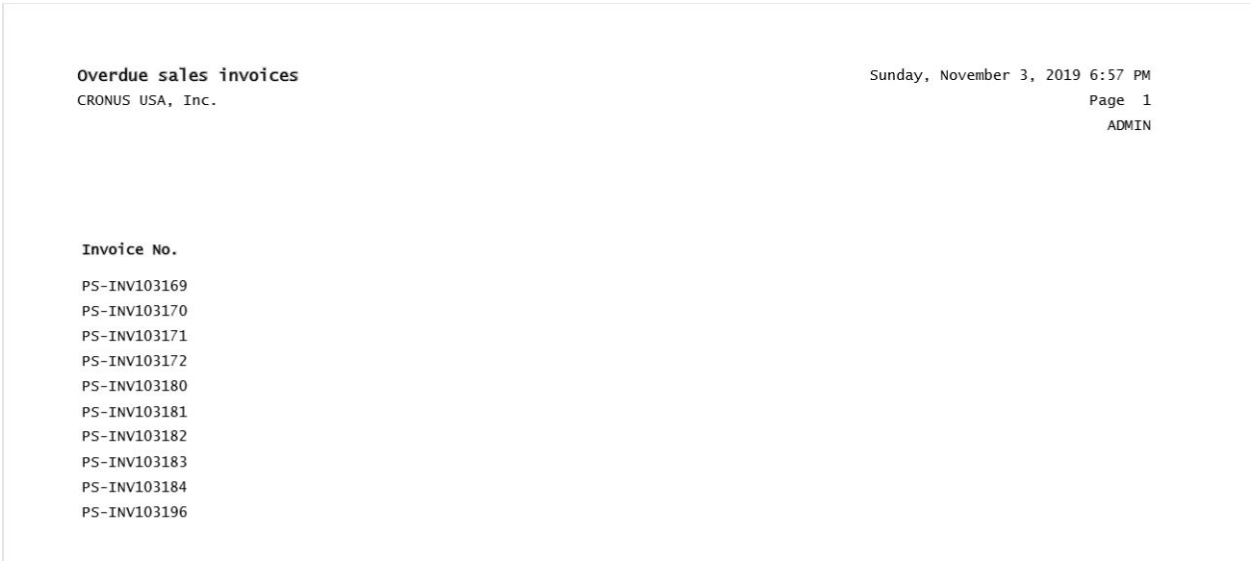

```
codeunit 50100 "ABS myCodeunit"
{
    1 reference
    procedure OverdueSalesInvoices(OverdueSalesInvList: List of [Code[20]])
    var
        CustomerLedgerEntry: record "Cust. Ledger Entry";
        InvoiceNo: code[20];
    begin
        CustomerLedgerEntry.SetCurrentKey("Document Type", Open);
        CustomerLedgerEntry.SetRange("Document Type", CustomerLedgerEntry."Document Type"::Invoice);
        CustomerLedgerEntry.SetRange(open, true);
        CustomerLedgerEntry.SetFilter("Due Date", '<%1', WorkDate());
        CustomerLedgerEntry.SetFilter("Remaining Amt. (LCY)", '<> 0');
        if CustomerLedgerEntry.FindSet() then
            repeat
                OverdueSalesInvList.Add(CustomerLedgerEntry."Document No.");
            until CustomerLedgerEntry.Next() = 0;

    end;
}
```

Voy a usar un reporte para mostrar los resultados, solo para que sean visualmente más claros.

```
43          trigger OnAfterGetRecord()
44          begin
45              InvoiceNo := OverdueSalesInvList.Get(Number);
46          end;
47
48          trigger OnPreDataItem()
49          begin
50              Mycodeunit.OverdueSalesInvoices(OverdueSalesInvList);
51              SetRange(Number, 1, OverdueSalesInvList.Count);
52          end;
```

Como puedes ver, a pesar de que el parámetro no fue pasado como var, la variable contiene los resultados esperados.



Overdue sales invoices
CRONUS USA, Inc.

Sunday, November 3, 2019 6:57 PM
Page 1
ADMIN

Invoice No.
PS-INV103169
PS-INV103170
PS-INV103171
PS-INV103172
PS-INV103180
PS-INV103181
PS-INV103182
PS-INV103183
PS-INV103184
PS-INV103196

Esto pasa porque el tipo Lista es tipo referencia, por lo tanto en ambos lugares, el reporte y el codeunit están apuntando al mismo objeto en memoria.

Note que si se usa el método **Clear() sobre la variable tipo lista, el sistema creará una nueva instancia de esa variable.

Acerca del Autor

María Helena Ocampo es una desarrolladora senior en Advanced Business Systems con más de 10 años de experiencia en las versiones entre Dynamics NAV 5.0 y Dynamics 365 Business Central.