User Defined Functions

- User-Defined **Procedures.**
- User-Defined **Functions**.

User Defined Functions

User-Defined Procedures

• with user defined functions, you can **break large script** tasks **into smaller ones**.

• Another term for user-defined functions is **subroutines** which, as the name implies, are pieces of script code which perform tasks within the main script.

• User-defined procedure subroutines are **the most common type of subroutine**.

• User-defined procedures are **declared after** the definition (**CONST, TYPE**, and **VAR**) blocks of a script, but before the script body.

• Just like a script, subroutines may have any of the **standard VectorScript definition blocks** (LABEL, CONST, TYPE, or VAR) as well as a script body.

• The general syntax for user-defined procedures is:

PROCEDURE <procedure identifier>[(<parameter list>)]
e.g. PROCEDURE SumOfSquares(limit:INTEGER;VAR result:INTEGER);

User Defined Functions

User-Defined Procedures

• Following the subroutine identifier is the **parameter list** for the subroutine. This optional list defines a method of **moving data in and out** of the subroutine.

• While it is possible to refer to values in the enclosing program blocks directly, doing so **would eliminate the ability** to easily use the subroutine in other code, which is one of the **major advantages** of using subroutines.

• The parameter list declares **a set of identifiers** (and their associated data types) that will be used **to pass data to and from the subroutine**.

• The **VAR keyword** indicates an identifier that will be **used to pass data** out of the subroutine to the calling code.

• **Identifiers** in the parameter list **can be treated as variables** and used within the subroutine script code.

• By calling the subroutine, **the order and types of the variable identifiers must exactly match those in the declaration.**

User Defined Functions

User-Defined Procedures

- example:

```
Procedure testSubroutine;

VAR
        myPosX,myPosY: REAL;
        myRadius, myDiameter            :INTEGER;

{SUBROUTINES}
Procedure CircelByPointAndRadius(rad, pointX, pointY : REAL; VAR diameter:INTEGER);
BEGIN
        diameter:=2*rad;
        oval(pointX-rad, pointY-rad, pointX+rad, pointY+rad);
END;
{End of declaration subroutine}

BEGIN;
        myPosX:= 23.5;
        myPosY:=myPosX;
        myRadius:= 50;
        CircelByPointAndRadius(myRadius, myPosX, myPosY, myDiameter);
END;
Run(testSubroutine);
```

User Defined Functions

User-Defined Functions

• User-defined functions incorporate **all the features of user-defined procedures**, but they have one additional feature which makes them extremely useful when writing scripts: **an associated value**.

• User-defined functions, unlike procedures, can **pass data out** of the subroutine through a return value, which associates the value with the subroutine identifier.

• User-defined function declarations have one additional requirement: **a return value type** after the parameter list. This data type indicates what type of data will be passed through the return value mechanism and will be associated with the identifier.

• The general syntax for user-defined functions is:

FUNCTION <procedure identifier>[(<parameter list>)]:<return value type>

User Defined Functions

User-Defined Functions

- example:

```
PROCEDURE SubrExample2;
VAR
        n,sum:INTEGER;

FUNCTION SumOfSquares(limit:INTEGER):INTEGER;
        BEGIN
                SumOfSquares:= limit*(limit+1)*(2*limit+1)/6;
        END;

BEGIN
        n:=IntDialog('Enter the limit value','0');
        {sum of squares for the first n integers}
        sum:= SumOfSquares(n);
        Message('The sum of squares is: ',sum);
END;
Run(SubrExample2);
```