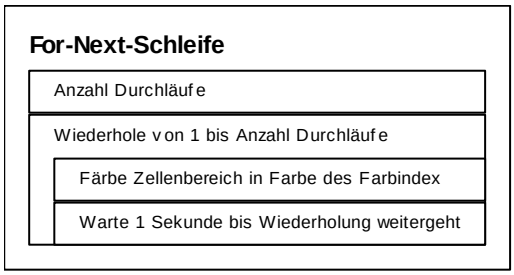
Grundstrukturen der Programmierung (4): Schleifenstrukturen (Iteration)

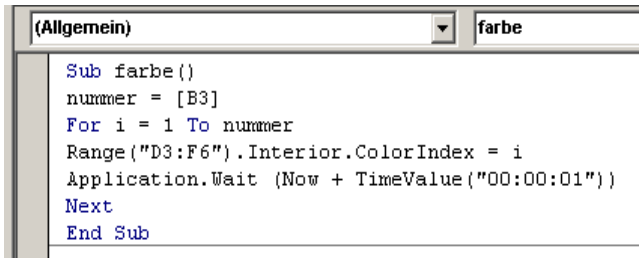
Mit Hilfe von Schleifenstrukturen lassen sich bestimmte Anweisungsfolgen mehrmals, abhängig von einer bestimmten Bedingung ausführen. Die Anzahl der **Schleifendurchläufe** wird durch den Wahrheitswert eines Ausdrucks - der **Schleifenbedingung** - oder durch den Wert eines numerischen Ausdrucks - des **Schleifenzählers** - definiert. Alle im Folgenden beschriebenen Schleifenkonstruktionen lassen sich, wenn erforderlich, ineinander verschachteln.

Zählergesteuerte Schleife (For – Next)

Die einfachste Schleifenform wird mit den Kommandos For und Next gebildet. Dabei wird einer Variablen zu Beginn der Schleife ein Startwert zugewiesen. Dieser Wert wird mit jedem Schleifendurchlauf automatisch erhöht, bis schließlich der Endwert erreicht ist. Diese Schleife kann nur eingesetzt werden, wenn vor Eintritt in die Schleife bereits die genaue Anzahl der Durchläufe feststeht. Sollte während der Durchläufe ein Ereignis eintreten, das den vorzeitigen Abbruch der Schleife erfordert, können Sie hierzu die Anweisung *Exit Do,* abhängig von einer Bedingung, benutzen. Die allgemeine Form der Anweisung lautet:

For Zähler = Anfangswert To Endwert [Step Schrittweite]
Anweisung(en)
[Exit Do]
Anweisung(en)
Next [Zähler]





Durch das optionale Schlüsselwort *Step* kann ein (positiver oder negativer) Wert angegeben werden, der mit jedem Schleifendurchlauf zur Schleifenvariablen addiert wird. Fehlt diese Angabe, wird immer um den Wert 1 weiter gezählt.

Schleifen mit Do – Loop

Steht nicht von vornherein fest, wie oft eine Schleife durchlaufen werden soll, so empfiehlt sich die *Do-Loop* – Schleife. Für diese Schleife existieren drei verschiedene Möglichkeiten zum Abbruch:

1) Die kopfgesteuerte Schleife

Do [While Bedingung]
Anweisung(en)
Loop

Hier wird die Abbruchbedingung am Schleifenkopf geprüft. Falls es möglich sein soll, dass die Schleife unter Umständen auch gar nicht durchlaufen werden soll, so muss diese Form gewählt werden.

2) Die fußgesteuerte Schleife

Do
Anweisung(en)
Loop [Until Bedingung]

In diesem Fall wird die Abbruchbedingung erst am Ende der Schleife geprüft, die Schleife wird also mindestens einmal durchlaufen. Somit ist diese Do-Loop-Variante vergleichbar mit der For-Next-Schleife da durch die folgenden Anweisungen die Schleife wenigstens einmal durchlaufen wird:

```
Do
   Zähler = Zähler + 1
   ……
Loop Until Zähler = 20
```

3) Die Schleife mit interner Abbruchbedingung

Hier wird an einer beliebigen Stelle innerhalb der (in ihrer Grundstruktur endlosen) Schleife eine Bedingung geprüft, die ggf. zum Abbruch führt. Davon abgesehen kann die *Exit*-Abbruchbedingung natürlich auch in kopf- oder fußgesteuerten zusätzlich eingesetzt werden.

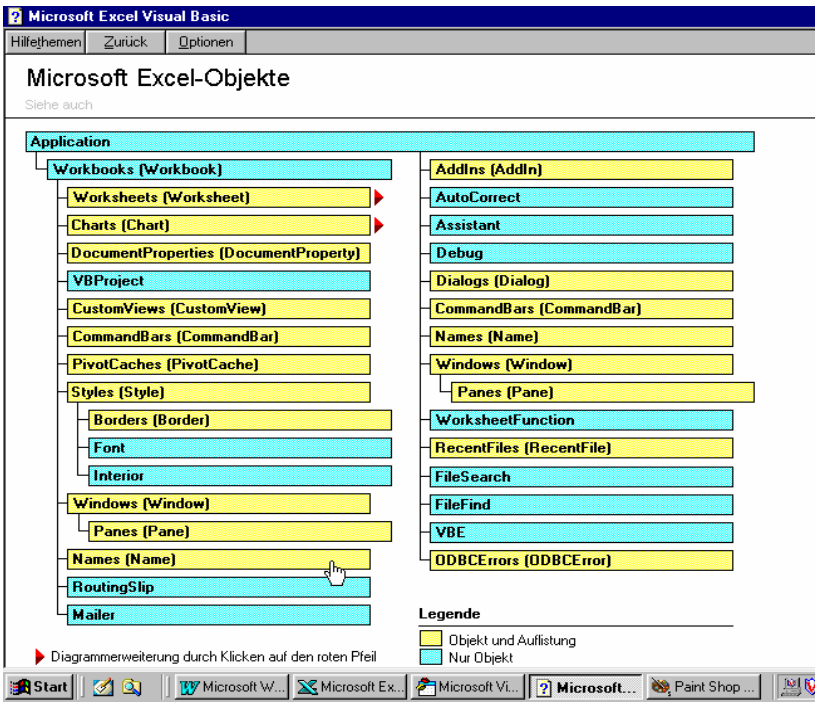
```
Do
   Anweisung(en)
   If Bedingung Then Exit
Loop
```

Ojekte – Methoden - Eigenschaften

Alle VBA Elemente von Excel (Access, Powerpoint, WinWord) werden als **Objekte** bezeichnet. Die Objekte sind hierarchisch organisiert. An oberster Stelle steht das Application – Objekt, welches die Excel-Anwendung selbst darstellt

Eigenschaften bestimmen die charakteristischen Merkmale eines Objektes, z. B. die Hintergrundfarbe einer Zelle, die Ausrichtung einer Tabellenzelle usw. In der Regel muss vor einer Eigenschaft das Objekt angegeben werden, auf das sich diese Eigenschaft bezieht.

Methoden ändern den Wert einer Objekteigenschaft und führen auch Aktionen auf oder mit Daten aus, die das Objekt speichert. Sie sind mit Makros oder Prozeduren vergleichbar, die an ein bestimmtes Objekt gebunden sind. Methoden ermöglichen den Zugriff auf andere Objekte, z. B. Sheets("blattname") bzw. Sheets(n).



Das Objekt "Auto" hat z.B. die Eigenschaft "Farbe" und "Antriebsart", als Methoden sind z.B. verfügbar "Fahren" oder "Parken". Eigenschaften können Ausprägungen haben, zum Beispiel Farbe = "Rot" oder Antriebsart = "Dieselmotor", Methoden können Argumente haben, z.B. Fahren "geradeaus". Die Methode "Fahren" wird z.B. ausgelöst durch das Ereignis "Gaspedal treten".

Das Objekt Excel-Zelle (RANGE()- oder CELLS()-Objekt) hat unter anderem die Eigenschaften "Hintergrundfarbe" oder "Schriftart", als Methode kann sie beispielsweise aktiviert oder verschoben werden. Eigenschaften und Methoden werden mit einem Punkt an das Objekt angehängt, so wie Unterobjekte an Objekte.

Allgemeine Syntax: Um die Eigenschaften oder Methoden eines Objektes zu nutzen, muss man das Objekt zusammen mit der gewünschten Eigenschaft oder Methode angeben. Objekte, Methoden und Eigenschaften werden durch einen **.** abgetrennt, z.B. Application.Sheets("Tabelle1").Activate

Nähere Erläuterungen zu allen in VBA für Excel verfügbaren Objekten und ihren Methoden und Eigenschaften finden Sie in der VBA-Hilfe. Sehr hilfreich ist auch die Eigenschaft des VBA-Editors, in den meisten Fällen nach der Eingabe eines Objektes und eines anschließenden Punktes eine Auswahl der für dieses Objekt verfügbaren Methoden und Eigenschaften sowie Unterobjekte zu präsentieren.

Aufgabenstellung:

1. Erstellen Sie die Funktion „Pruef“, die eine Prüfziffer für die Artikelnummern erstellt. Das Produkt der gegebenen Zahlen soll mit dem Restwertverfahren 7 (Modula 7) die Prüfziffer ergeben. Beispiel: P1234 ergibt 1 x 2 x 3 x 5 = 30 / 7 = 4 Rest 2. ***Prüfziffer ist die 2!*** Setzen Sie diesen Fall als For-Next und als Do-Loop-Schleife um. Zur Stringverarbeitung recherchieren Sie bitte nach der Funktion MID sowie dem Operator MOD in der VBA-Hilfe.
2. Erstellen Sie eine Prozedur für das Tabellenblatt „Lager“, dass aus einer aktuellen Zelle (activecell) heraus ermittelt, wie viele nachfolgende Datensätze vorhanden sind. Lassen Sie Anzahl der Datensätze über eine Messagebox ***MSBGOX*** ausgeben

Tipps:

- Über die Eigenschaften (activecell).ROW [=Zeile] sowie (activecell).COLUMN [=Spalte] erhalten Sie die aktuelle Position des Objekts CELLS. [CELLS(3,4) ist die Zelle D3].
- Durch Einsatz der Variablen i und j kann das CELLS-Objekt erweitert werden. [CELLS(i,j)]

3. Erstellen Sie aus allen Funktionen bzw. aus der Prozedur ein Struktogramm.

Datensätze zählen

Start = ActiveCell.Row
i = Start
j = ActiveCell.Column

springe eine Zelle tiefer
(i = i + 1)

wiederhole bis die erste
leere Zelle gefunden wurde

Ausgabe:

Es sind noch

i - Start - 1

Datensätze vorhanden

Prüefziffer (For Next)

Wertübergabe (Variable WERT an Funktion)

pruef = 1

For i = 2 to 5

pruef = pruef * i-te Stelle
(Teil bzw. MID)

Ergebnis der Funktion:

pruef = Restwert 7 (pruef)
(MOD 7)

Prüfziffer (Do Loop)

Wertübergabe (Variable WERT2 an Funktion)

pruef2 = 1

Zähler erhöhen

(i = i +1)

pruef 2 = pruef2 x i-te Stelle

Wiederhole bis i = 5

Ergebnis der Funktion:

pruef2 = Restwert 7 (pruef2)