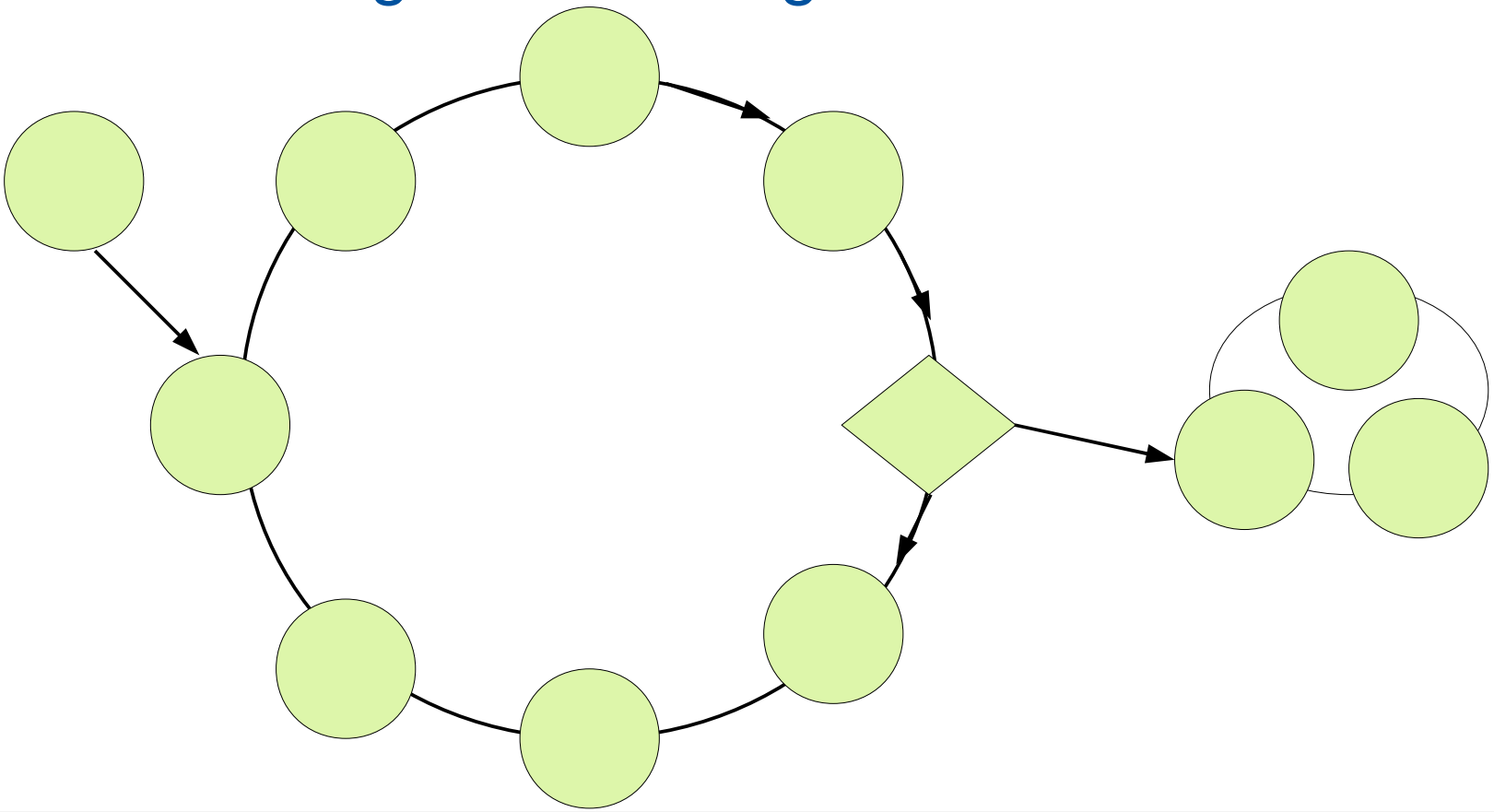
Excel – VBA
Bedingte Anweisungen und Schleifen



Algorithmus

- Genau definierte Verarbeitungsvorschrift zur Lösung einer Aufgabe.
- Eine Folge von Arbeitsschritten zur Lösung eines Problems.
- Endliche Folge von Anweisungen, die nacheinander ausgeführt werden.
- Beispiele aus dem täglichen Leben:
  - Kochrezepte
  - Steuerprogramme für technische Geräte (zum Beispiel Waschmaschinen, Mikrowelle)

Beispiel: Bedienung einer Waschmaschine

- Tür der Waschmaschine öffnen.
- Max. 5 kg Wäsche (einer Farbe ☺) einfüllen.
- Tür der Waschmaschine schließen.
- Waschmittel passend zur Farbe der Wäsche in die kleine Schublade für den Hauptwaschgang füllen.
- Wasserzulauf öffnen.
- Waschprogramm wählen.
- Starttaste drücken.
- Waschvorgang abwarten.
- Nach Programm-Ende Maschine abstellen.
- Wasserzulauf schließen.
- Tür öffnen und Wäsche entnehmen.

Eigenschaften eines Algorithmus

- Ein Algorithmus benötigt endlich viele Arbeitsschritte.
- Ein Algorithmus ist beschreibbar.
- Jeder Arbeitsschritt ist ausführbar.
- Ein Algorithmus liefert unter identischen Startbedingungen immer das gleiche Endergebnis.
- Der Ablauf des Verfahrens ist eindeutig definiert.

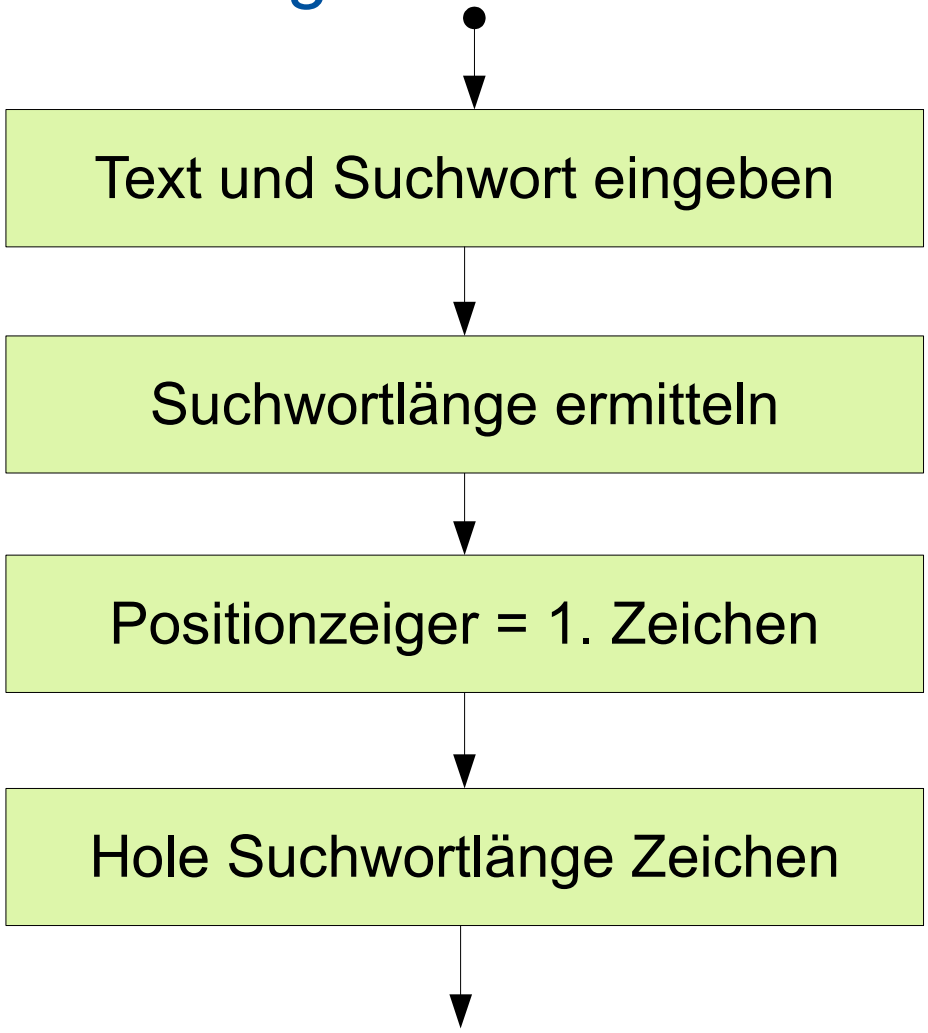
Beispiel

- Aufgabe: Suche ein bestimmtes Wort in einen Text.
- Eingangsdaten:
  - Variable text: Zeichenfolge (String), in der ein bestimmtes Wort gesucht werden soll.
  - Variable wort: Zeichenfolge, die gesucht wird.
- Ausgangsdaten:
  - Ist das Wort in der Zeichenfolge text vorhanden?

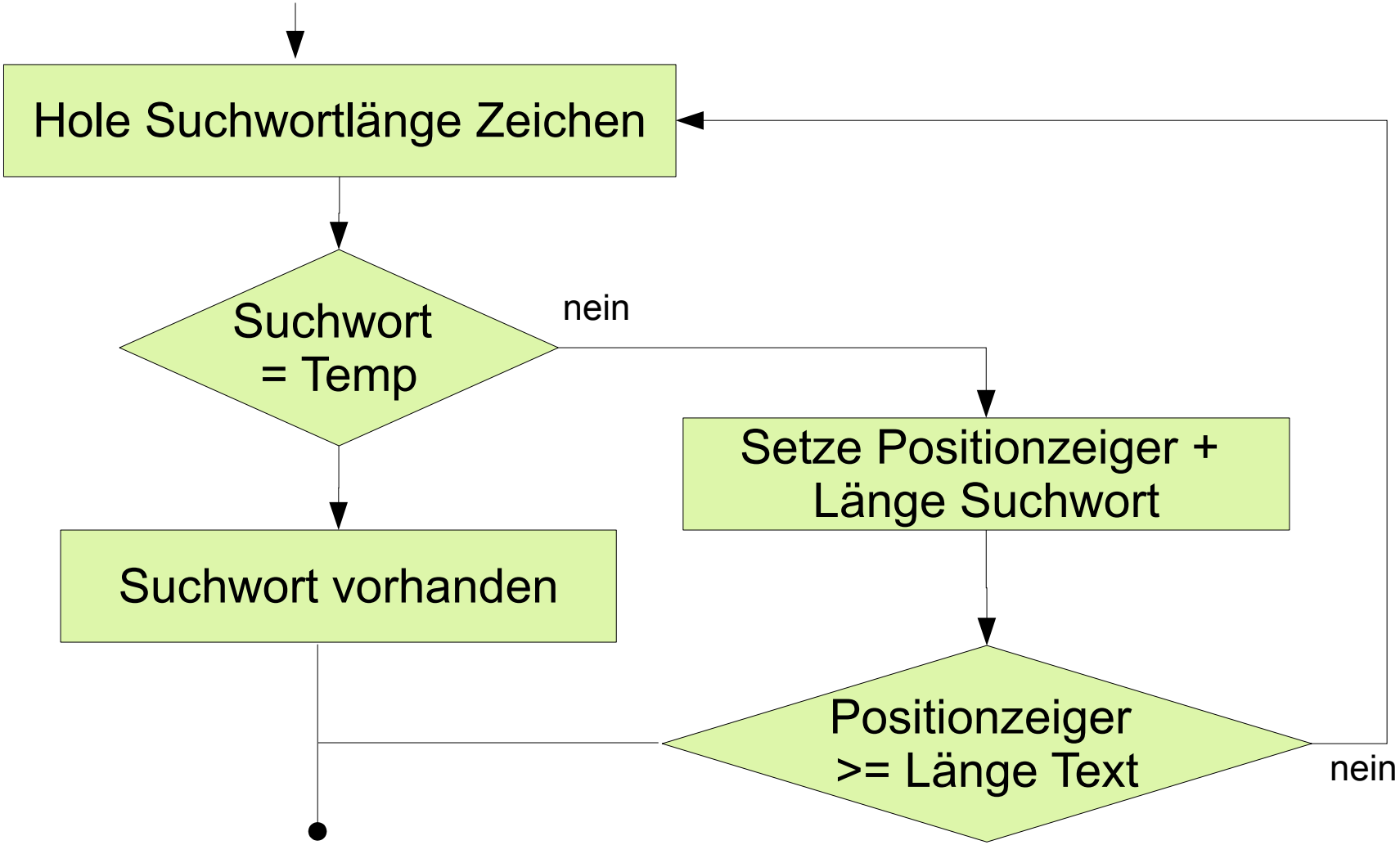
Grafische Darstellung

- .. in einem Flussdiagram:
  - Ein Flussdiagramm ist ein Ablaufdiagramm für Computerprogramme.
  - Die benutzten Symbole sind in der DIN 66001 genormt.
- ... in einem Struktogramm:
  - Struktogramme werden auch als Nassi-Shneidermann-Diagramme bezeichnet.
  - Die benutzten Symbole sind in der DIN 66261 genormt.
  - Struktogramm-Editoren sind unter http://de.wikipedia.org/wiki/Nassi-Shneiderman-Diagramm zu finden.

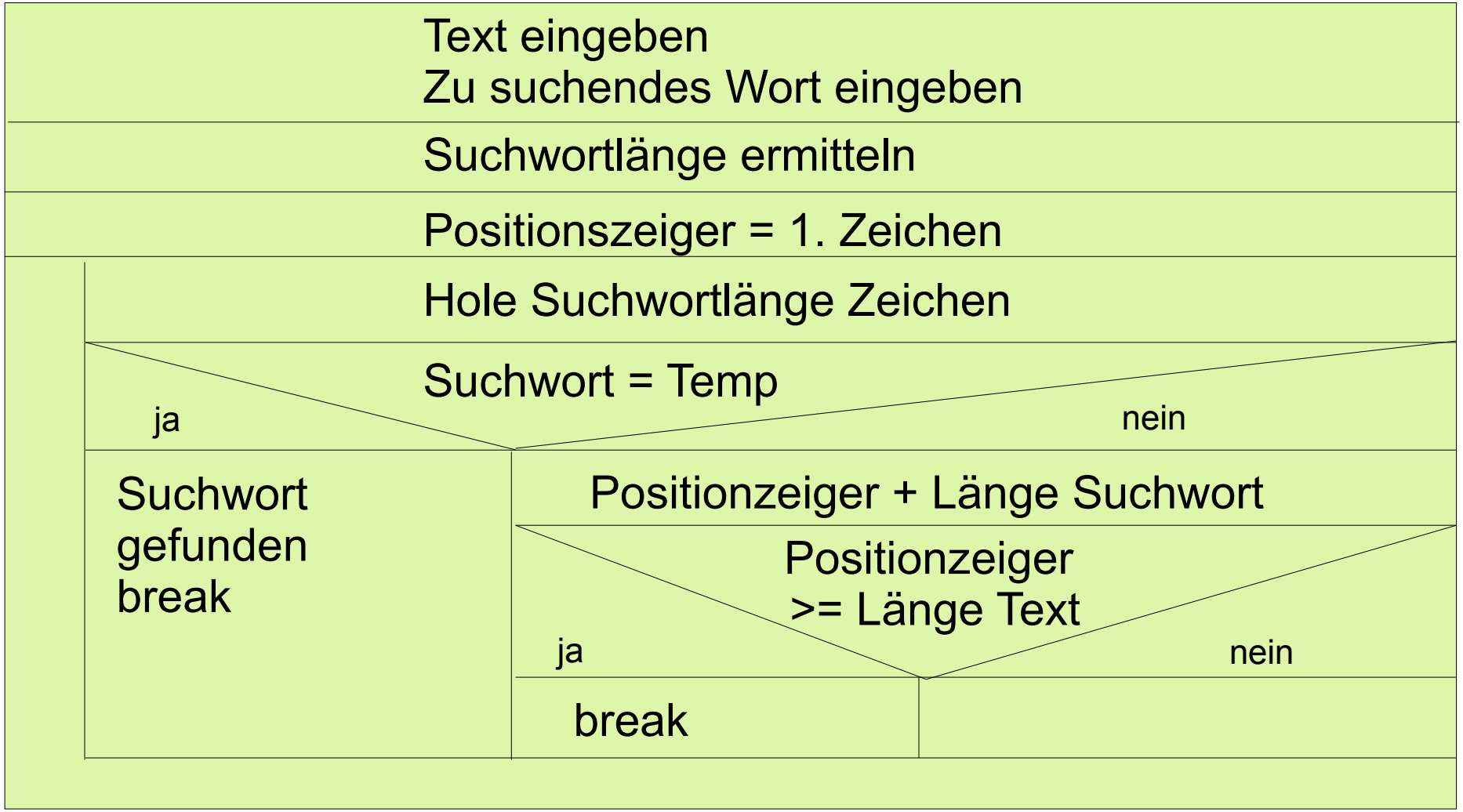
Flussdiagramm: Wort-Suche



Flussdiagramm: Wort-Suche



Struktogramm: Wort-Suche



Vorteile von Struktogrammen

- Struktogramme erzwingen einen sequentiellen Programmablauf ohne Sprünge.
- Im Allgemeinen führt die Verwendung von Struktogrammen dazu, dass Programme besser strukturiert sind, als bei der Verwendung von Programmablaufplänen.
- Verwenden Sie vorzugsweise (zumindest während Sie ihre erste Programmiersprache lernen) Struktogramme.
- Bevor Sie mit der Codierung beginnen, machen Sie einen Programmentwurf.

Anweisung

- ... bestehen aus VBA-Schlüsselwörtern sowie aus Operatoren und Operanden.
- ... beschreiben die einzelnen Arbeitsschritte in einem Algorithmus.
- ... beschreiben Aktionen, die der Computer ausführen soll.
- ... können in Abhängigkeit einer Bedingungen durchlaufen werden.
- ... können unter bestimmten Bedingungen wiederholt werden.

Kontrollanweisungen

- Mit Hilfe von Kontrollstrukturen kann der Programmablauf beeinflusst werden.
- In Abhängigkeit vom Wert einer oder mehrerer Variablen wird der Programmablauf gesteuert.
- Es gibt folgende Möglichkeiten:
  - Auswahlanweisungen: In Abhängigkeit eines Kriteriums werden Anweisungen ausgeführt oder nicht.
  - Schleifen (Iterationsanweisungen): Anweisungen werden wiederholt ausgeführt.

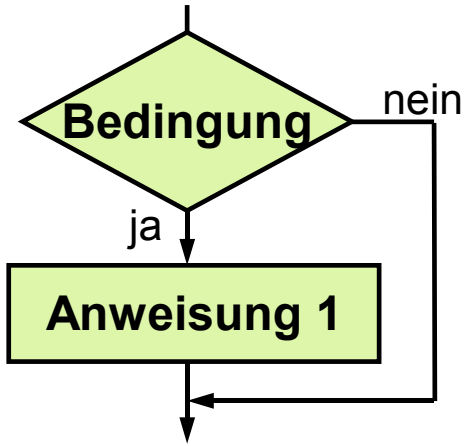
Auswahlanweisungen

- if – else – Anweisungen. In Abhängigkeit einer Bedingung werden Anweisungen ausgeführt oder nicht.
- select case – Anweisungen. Der Wert eines Ausdrucks wird ausgewertet. In Abhängigkeit des Wertes wird zu den verschiedenen Fällen verzweigt.

Bedingte Anweisung

- .. oder Selektionsanweisungen.
- Wenn – Dann – Anweisung.
- ... beginnen mit If (wenn) und enden mit End If.
- Zwischen diesen beiden Schlüsselwörtern stehen Anweisungen.
- Diese Anweisungen werden nur ausgeführt, wenn die Bedingung erfüllt ist.
- Die Bedingung folgt dem If und endet mit Then (Dann).
- Falls die Bedingung nicht erfüllt ist, werden die Anweisungen nach dem End If ausgeführt.

```
If (bedingung) Then
    anweisung1
    anweisung2
End If

anweisung
```



Beispiel: if-Anweisung

```
Private Sub Worksheet_Activate()
    Dim wert As Integer

    wert = Range("A14").Value

    If wert > 0 Then
        Range("A14").NumberFormatLocal = "00,00"
        Range("A14").Font.Color = vbBlack
        Range("B14").Value = wert / 100
    End If
End Sub
```

Bedingungen

- ... sind Ausdrücke, die einen boolschen Wert zurückliefern. Ein boolscher Wert kennt nur zwei Zustände: true (wahr) oder false (falsch).
- ... vergleichen mit Hilfe von bestimmten Operatoren zwei Werte.
- ... sind zum Beispiel:
  - Wenn die Bestellmenge eine gewisse Höchstmenge überschreitet...
  - Wenn der Kontostand dem Dispo entspricht...
  - Wenn die Strecke A doppelt so lang ist wie Strecke B...
  - Wenn die Warenmenge eine Mindestmenge unterschreitet...

Vergleichsoperatoren

Operator	Beschreibung	Beispiel	Ergebnis
=	gleich	3 = 2	false
<>	ungleich	3 <> 2	true
<	kleiner als	3 < 2	false
<=	kleiner gleich	3 <= 2	false
>	größer	3 > 2	true
>=	größer gleich	3 >= 2	true

Methoden für einen String-Vergleich

```
Sub StringVergleich()
    Const txtString As String = "Schneekönigin"
    Const txtVergleich As String = "Schnee"

    Dim boolResult As Boolean
    Dim intResult As Integer

    intResult = StrComp(txtString, txtVergleich, vbBinaryCompare)

    boolResult = txtString Like txtVergleich & "*"

    boolResult = (txtString = txtVergleich)
End Sub
```

String Compare

StrComp(txtString, txtVergleich, vbBinaryCompare)

- ... vergleicht zwei Strings in Abhängigkeit einer Vergleichsart.
- ... liefert einen Integer-Wert zurück.
  - 0. Die Strings sind gleich.
  - -1. Der erste String liegt in der ASCII-Tabelle vor dem zweiten String.
  - 1. Der zweite String liegt in der ASCII-Tabelle vor dem ersten String.
  - Null. Eine der beiden Zeichenketten ist Null.

Vergleichsarten

- Eine Vergleichsart
  - ... kann als Parameter an eine Funktion übergeben werden.
  - ... kann als Anweisung am Anfang eines Moduls angegeben werden.
- Option Compare Binary vbBinaryCompare
  - Standardeinstellung.
  - Vergleich in Abhängigkeit der ASCII-Tabelle.
  - Groß- und Kleinschreibung wird unterschieden.
- Option Compare Text vbTextCompare
  - Textbasierter Vergleich.
  - Groß- und Kleinschreibung wird nicht unterschieden.

String mit Hilfe des Gleichheitszeichen vergleichen

boolResult = (txtString = txtVergleich)

- Zwei Strings werden Zeichen für Zeichen verglichen.
- Die Zeichen werden in Abhängigkeit der ASCII-Tabelle verglichen.
- Die Lesbarkeit des Ausdrucks wird durch die Klammerung erhöht.
- Eine Nutzung von Platzhaltern für einzelne Zeichen ist nicht möglich.

Vergleichsoperator Like für Strings

```
boolResult = txtString Like txtVergleich & "*"
```

- Zwei Zeichenketten werden in Abhängigkeit der Option Compare-Anweisung miteinander verglichen.
- Es können Platzhalter für einzelne Zeichen genutzt werden.

Platzhalter in Strings

- Das Fragezeichen ersetzt ein Zeichen.
- Das Sternchen ersetzt kein oder mehrere Zeichen.
- Das Hash-Zeichen (#) ersetzt ein nummerisches Zeichen (0..9).
- [Startzeichen - Endezeichen] nutzt einen Zeichenraum als Platzhalter.
- [Zeichen01, Zeichen02, ...] nutzt eine Liste von erlaubten Zeichen als Platzhalter.
- Ein Ausrufezeichen invertiert die angegebenen Werte. Die Zeichen dürfen nicht im Wort enthalten sein.

Beispiele

Vergleich	Beispiele für ein Resultat = True
Like "World"	World
Like "M*"	Meier Meierei
Like "M*er"	Meier Maurer
Like "M??er"	Mauer Meier
Like "1-?-*"	1-A-1234 1-1-Ananas
Like "30###"	30159 30456

Beispiele

Vergleich	Beispiele für ein Resultat = True
Like "B[a, u]ch"	Bach Buch
Like "3015[5-9]"	30159 30156
Like "[!0-9]-###-#####"	D-123-45674 E-789-45615

Logische Operatoren nutzen

```
Dim result As Boolean
Dim wert As Integer

wert = 4
result = (wert < 5) And (wert > 6)   ' (4 < 5) UND (4 > 6) = Falsch

result = (wert < 5) Or (wert > 6)    ' (4 < 5) ODER (4 > 6) = RICHTIG

result = Not (wert = 5)              ' NICHT(4 = 5) = WAHR
result = (wert <> 5)                 ' (4 ungleich 5) = WAHR
```

Operatoren für die Verknüpfung

- AND. Und. Konjunktion. Alle Bedingungen müssen wahr sein.
- OR. Oder. Disjunktion. Eine der Bedingungen muss wahr sein.
- NOT. Nicht. Negation. Invertiert die Bedingung.
- XOR. Antivalenz. Zwei Bedingungen liefern ein unterschiedliches Ergebnis. Wenn die eine Bedingung true liefert, muss die andere Bedingung false liefern.
- EQL. Äquivalenz. Zwei Bedingungen liefern ein gleiches Ergebnis. Wenn die eine Bedingung true liefert, muss die andere Bedingung auch true liefern.
- IMP. Implikation. Die zweite Bedingung muss nur wahr sein, wenn die erste Bedingung auch wahr ist. Einer falschen Bedingung können auch richtige Bedingungen folgen.

Wahrheitstabelle

Bedingung		Möglichkeiten		
a	b	NOT(a)	a AND b	a OR b
false	false	true	false	false
true	false	false	false	true
false	true	true	false	true
true	false	false	false	true

Beispiel: Verknüpfung mit AND

```
Dim bestellpreis As Double
Dim rabatt As Double
Dim preis As Double
Dim rabattpreis As Double
Dim bestellmenge As Integer

bestellPreis = (bestellmenge * preis)
rabatt = 0

If (bestellmenge > 100) And (bestellmenge < 500) Then
    rabatt = 0.03
    rabattpreis = bestellpreis * rabatt
    bestellpreis = bestellpreis - rabattpreis
End If
```

Beispiel: Verknüpfung mit OR

```
Dim zahl As Double
Dim mass As Char
Dim umrechnung As Double

mass = "g"

If ((mass = "m") Or (mass = "g")) Then

    umrechnung = zahl / 1000
End If
```

Beispiel: Not nutzen

```
Dim zahlA As Integer
Dim zahlB As Integer
Dim result As Integer

If Not (zahlB >= 0) Then
    MsgBox "Division durch Null nicht erlaubt"
Else
    result = zahlA \ zahlB
End If
```

Andere Möglichkeiten

```
Dim zahlA As Integer
Dim zahlB As Integer
Dim result As Integer

If (zahlB < 0) Then
    MsgBox "Division durch Null nicht erlaubt"
Else
    result = zahlA \ zahlB
End If

If (zahlB <> 0) Then
    result = zahlA \ zahlB
Else
    MsgBox "Division durch Null nicht erlaubt"
End If
```

Hinweise

- Beispiel: (var1 != var2) AND (var2 > 10)
  - Zuerst wird die linke Bedingung (var1 != var2) ausgewertet.
  - Anschließend wird die rechte Bedingung (var2 > 10) ausgewertet.
- Ein Ausdruck wird immer von links nach rechts ausgewertet!
- Sie können die Operatoren beliebig oft in beliebiger Mischung in einer Bedingung nutzen.
- Um die Lesbarkeit zu erhöhen, sollten die verschiedenen Elemente der Bedingung mit runden Klammern zusammengefasst werden.
- Falls verschiedene Operatoren gemischt werden, muss die Bindung der Operatoren beachtet werden.

Rangfolge der Operatoren

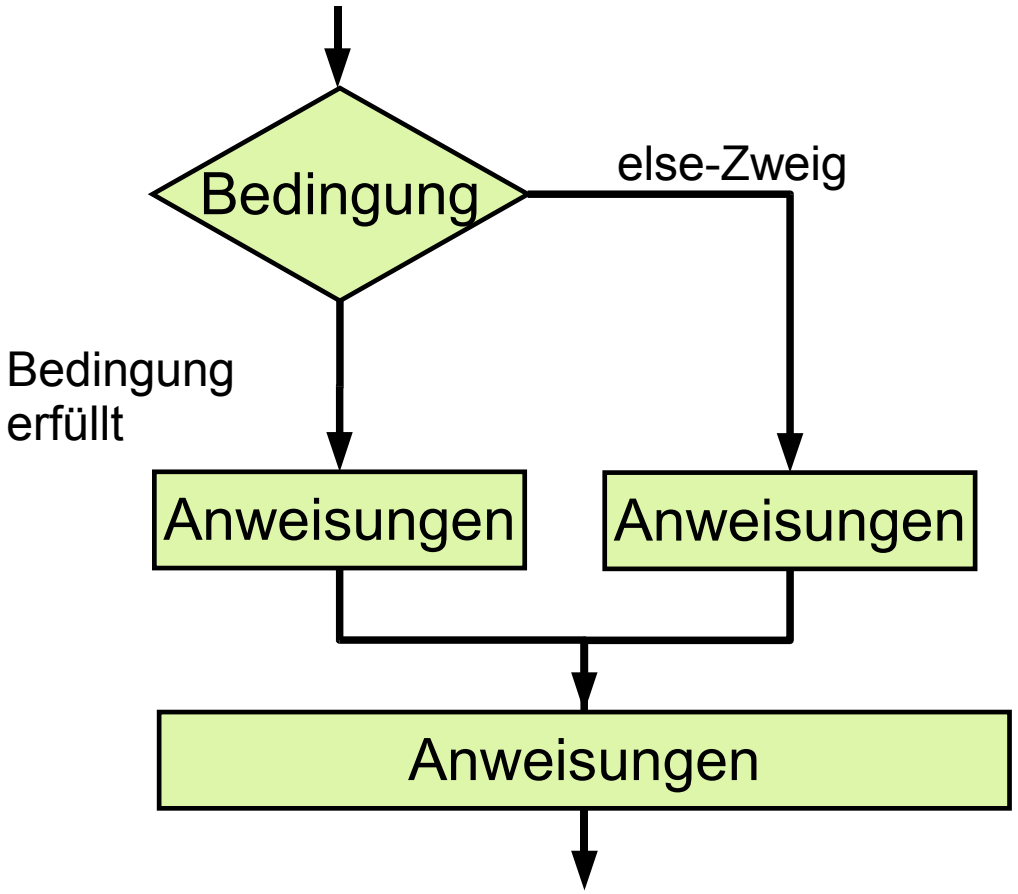
(	)					
^						
- (Vorzeichen)						
*	/					
\						
Mod						
+	-					
&						
=	<>	<	>	<=	=>	Like
NOT						
AND						
OR						

if-else-Anweisung

- Wenn – Dann – Andernfalls – Anweisung.
- Falls die Bedingung nicht erfüllt wird, werden die Anweisungen im Else-Zweig ausgeführt.
- Der Else-Zweig
  - ... kann nie ohne eine If-Anweisung existieren.
  - ... hat die gleiche Einrücktiefe wie die dazugehörige If-Anweisung.
  - ... benötigt keine Bedingung.
  - ... beschreibt den Standardfall.
  - ... fängt alle nicht behandelten Fälle ab.

```
If (bedingung) Then
    anweisung1
    anweisung2
Else
    anweisung1
    anweisung2
End If

anweisung
```

Aufbau der bedingten Anweisungen



Beispiel: else-Anweisung

```
Private Sub Worksheet_Activate()
  Dim wert As Integer

    wert = Range("A14").Value

    If wert > 0 Then
        Range("A14").NumberFormatLocal = "00,00"
        Range("A14").Font.Color = vbBlack
        Range("B14").Value = wert / 100
    Else
        Range("A14").Font.Color = vbBlack
        Range("A14").NumberFormat = "@"
        Range("B14").Value = "Kein Eintrag"
    End If
End Sub
```

Verschachtelte Anweisungen

```
result = zahlA Mod zahlB

If (result = 0) Then

    If (zahlA = zahlB) Then
        MsgBox "Die Zahlen sind gleich."
    Else
        MsgBox "Division ohne Rest"
    End If

Else
    MsgBox "Rest der Division: " & result
End If
```

Hinweise

- Jede Ebene einer verschachtelten if-Anweisung wird mit Hilfe des Tabulators eingerückt.
- Um die Übersicht zu behalten, sollten Sie nie mehr als 5 Ebenen tief verschachteln.
- Vermeiden Sie unterschiedliche Datentypen auf beiden Seiten eines Vergleichsoperators.
- Bei Dezimalzahlen sollte ein Test auf Gleichheit vermieden werden.

Mehrstufige if-else-Anweisungen

```
Dim masse As Char
Dim einheit As String

 If (masse = "m") Then
      einheit = "Meter"

 ElseIf (masse = "cm") Then
       einheit = "Zentimeter"

 ElseIf (masse = "mm") Then
      einheit = "Milimeter"

Else
     einheit = "Nicht bekannt"
 End If
```

Select Case-Anweisungen

```
Select Case masse
    Case "m"
        einheit = "Meter"
    Case "cm"
        einheit = "Zentimeter"
    Case "mm"
        einheit = "Milimeter"
    Case Else
        einheit = "Nicht bekannt"
End Select
```

Erläuterung

- Select Case leitet eine Fallunterscheidung ein.
- Dem Schlüsselwort Select Case folgt eine Variable, die in dem Block untersucht wird.
- Case leitet den einzelnen Fall ein. Entspricht der zu überprüfende Variablenwert dem angegebenen Wert, werden alle zu dem Fall gehörenden Anweisungen ausgeführt.
- Eine Select Case-Anweisung überprüft standardmäßig einen Ausdruck auf Gleichheit!

```
Select Case variable
      Case "fall1"
            anweisung11
      Case "fall2"
            anweisung21
      Case Else
            anweisung
End Select
```

Standardfall

- Case Else
  - ... beschreibt den Standardfall.
  - ... wird genutzt, wenn alle anderen beschriebenen Fälle nicht zutreffen.
  - Die Anweisung ist optional.

```
Select Case variable
      Case "fall1"
            anweisung11
      Case "fall2"
            anweisung21
      Case Else
            anweisung
End Select
```

Aufzählungen nutzen

```
Dim einkauf As String = "Bananen"
Dim kategorie As String

 Select Case einkauf

    Case "Bananen", "Äpfel"
       kategorie = "Obst"
    Case "Möhre", "Kohlrabi"
       kategorie = "Gemüse"
    Case "Basilikum", "Schnittlauch"
       kategorie = "Kräuter"

    Case Else
       kategorie = ""
 End Select
```

Der Wert der Variablen muss einem Element aus der Auflistung entsprechen. Die Auflistung kann aus beliebig vielen Werten bestehen.

Unter- und Obergrenzen nutzen

```
Dim bestellsumme As Double

Dim rabatt As Double

Select Case bestellsumme

    Case 500 To 1000
        rabatt = 0.2
    Case 1000 To 1500
        rabatt = 0.3
    Case Is > 1500
        rabatt = 0.4

    Case Else
        rabatt = 0
End Select
```

Untergrenze To Obergrenze. Der Wert der Variablen muss innerhalb der Grenzen liegen.

Vergleichsoperatoren nutzen

```
Dim bestellsumme As Double
Dim rabatt As Double

Select Case bestellsumme

    Case Is > 500, Is <= 1000
        rabatt = 0.2
    Case Is > 1000, Is <= 1500
        rabatt = 0.3
    Case Is > 1500
        rabatt = 0.4

    Case Else
        rabatt = 0
End Select
```

Is ist ein Synonym für die zu untersuchende Variable. Falls nicht auf Gleichheit getestet wird, muss das Schlüsselwort Is genutzt werden. Verschiedene Vergleiche können in einer Liste zusammengefasst werden. Die einzelnen Vergleiche werden durch Kommata getrennt.

Logische Operatoren nutzen

```
Dim bestellsumme As Double
Dim rabatt As Double

Select Case bestellsumme

    Case Is > 500 And bestellsumme <= 1000
        rabatt = 0.2
    Case Is > 1000, Is <= 1500
        rabatt = 0.3
    Case Is > 1500
        rabatt = 0.4

    Case Else
        rabatt = 0
End Select
```

Bei einer Verkettung von Vergleichen mit Hilfe von logischen Operatoren muss die zu vergleichende Variable wiederholt werden.

Schleifen (Iterationsanweisungen)

- ... führen Anweisungen mehrfach aus. Die Anzahl der Wiederholungen muss nicht vorher festgelegt werden.
- ... können endlos laufen.
- Schleifen können vom Programmierer vorzeitig abgebrochen werden.
- Schleifen können verschachtelt werden.

Schleifen-Typen

- Zählschleifen
  - Die Anzahl der Durchläufe ist bekannt.
  - Mit Hilfe einer Variablen werden die Schleifendurchläufe gezählt und nach einer bestimmten Anzahl abgebrochen.
- Kopfgesteuerte und fußgesteuerte Schleifen
  - ... werden in Abhängigkeit einer Bedingung durchlaufen.
  - Die Anzahl der Durchläufe ist nicht bekannt.
  - ... werden mit Hilfe einer Bedingung gesteuert, die im Kopf oder Fuß der Schleife steht.
  - Falls die Bedingung erfüllt ist, werden die Anweisungen abgearbeitet.
  - Eine fußgesteuerte Schleife wird unabhängig von der Bedingung mindestens einmal durchlaufen.

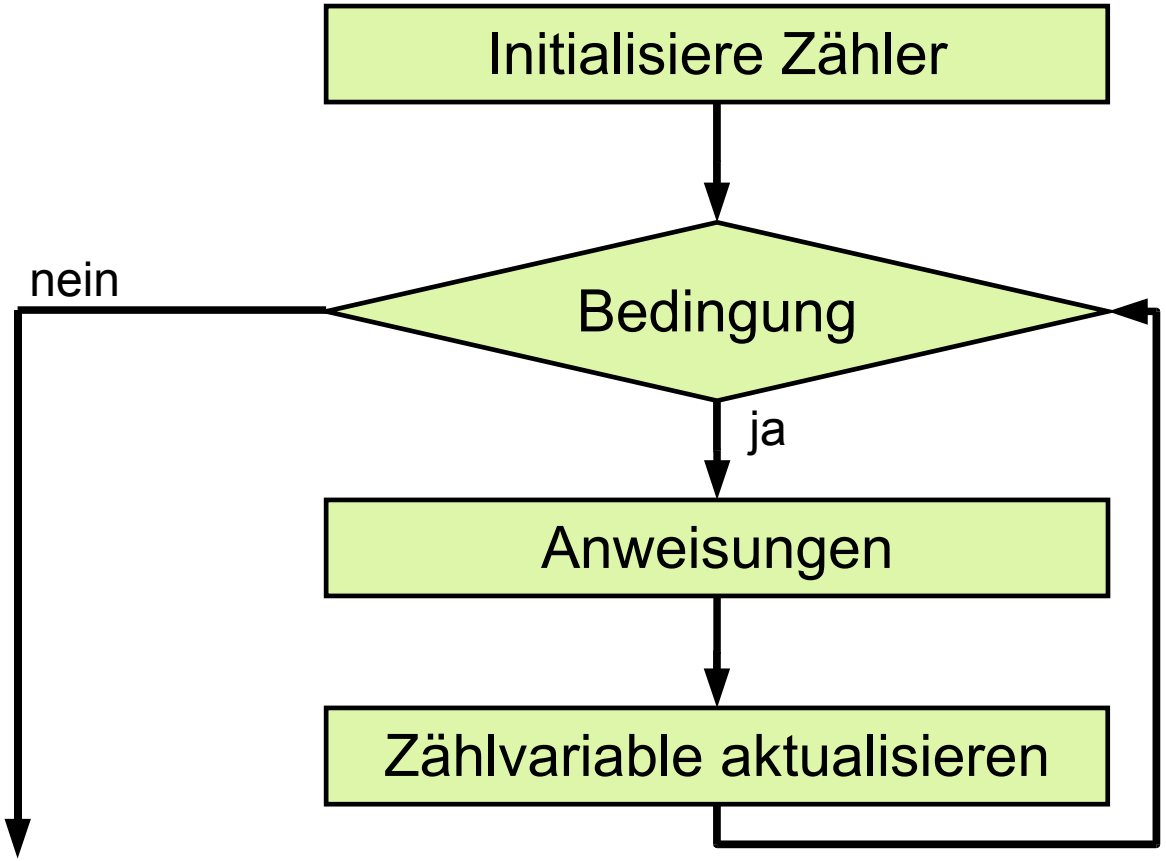
Zählschleife

```
Dim nMin As Integer
Dim nMax As Integer
Dim nCount As Integer
Dim nSumme As Integer

nMin = 1
nMax = 10
nSumme = 0

For nCount = nMin To nMax
     nSumme = nSumme + nCount
    Debug.Print "+ " & nCount & "= " & nSumme
Next nCount
```

Schleifenaufbau

- Der Schleifenkopf einer Zählschleife beginnt mit dem Schlüsselwort `For`.
- Im Schleifenkopf wird
  - ... der Zähler initialisiert (`nCount = nMin`). Die Zählvariable kann von einem beliebigen Typ sein.
  - ... die Anzahl der Durchläufe durch die Angabe einer Unter- und Obergrenze festgelegt (`nMin To nMax`).
- Die Schleife endet mit `Next nCount`.
- Im Schleifenfuß wird der Wert des Zähler um eins erhöht.

Arbeitsweise



Schrittweite verändern

```
Dim nMin As Integer
Dim nMax As Integer
Dim nCount As Integer
Dim nSumme As Integer

nMin = 1
nMax = 10
nSumme = 0

For nCount = nMin To nMax Step 2
     nSumme = nSumme + nCount
    Debug.Print "+ " & nCount & "= " & nSumme
Next nCount
```

Negative Werte nutzen

```
Dim nMin As Integer
Dim nMax As Integer
Dim nCount As Integer
Dim nSumme As Integer

nMin = 10
nMax = 1
nSumme = 0

For nCount = nMin To nMax Step -2
     nSumme = nSumme + nCount
    Debug.Print "+ " & nCount & "= " & nSumme
Next nCount
```

Die Obergrenze muss kleiner als die Untergrenze sein. Andernfalls wird die Schleife nicht durchlaufen.

Dezimalwerte nutzen

```
Dim nMin As Integer
Dim nMax As Integer
Dim nCount As Single
Dim nSumme As Single

nMin = 1
nMax = 10
nSumme = 0

For nCount = nMin To nMax Step 0.5
     nSumme = nSumme + nCount
    Debug.Print "+ " & nCount & "= " & nSumme
Next nCount
```

Der Zähler und die Schrittweite müssen den gleichen Datentyp besitzen. Bei unterschiedlichen Datentypen wird die Schleife nicht durchlaufen.

Schleifen verschachteln

```
For wert = 1 To 5
    ergebnis = 0

    For faktor = 0.5 To 2 Step 0.5
        ergebnis = faktor * wert

        If (faktor > 1.5) AND (wert = 2) Then
            Exit For
        End If
    Next faktor

    Debug.Print ergebnis
Next wert
```

Die innere Schleife wird vollständig abgearbeitet. Anschließend wird der nächste Schleifendurchlauf der äußeren Schleife gestartet.

Do... Loop -Schleifen

```
Do
    Anweisung
Loop
```

- ... beginnen mit Do und enden mit Loop.
- Die Schleife hat keine Bedingung. Die Schleife läuft endlos.

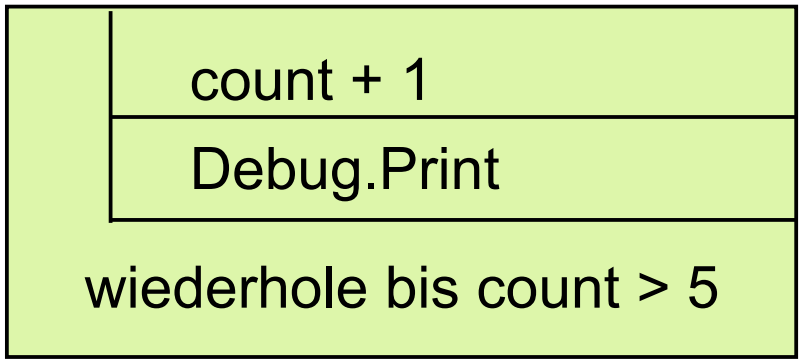
Fußgesteuerte Schleifen

- Die Abbruchbedingung steht im Fuß der Schleife.
- Die Anweisungen der Schleife werden einmal abgearbeitet. Anschließend wird die Bedingung überprüft.
- Bedingung mit Hilfe von Until (bis):
  - Die Schleife läuft solange bis die Bedingung erfüllt ist.
  - Sobald die Bedingung erfüllt ist, wird die Schleife abgebrochen.
- Bedingung mit Hilfe von While (während, so lange):
  - Die Schleife läuft solange wie die Bedingung erfüllt ist.
  - Sobald die Bedingung nicht erfüllt ist, wird die Schleife abgebrochen.

Do-Loop Until

```
Dim count As Integer = 0

Do
    count = count + 1
    Debug.Print count
Loop Until count > 5
```

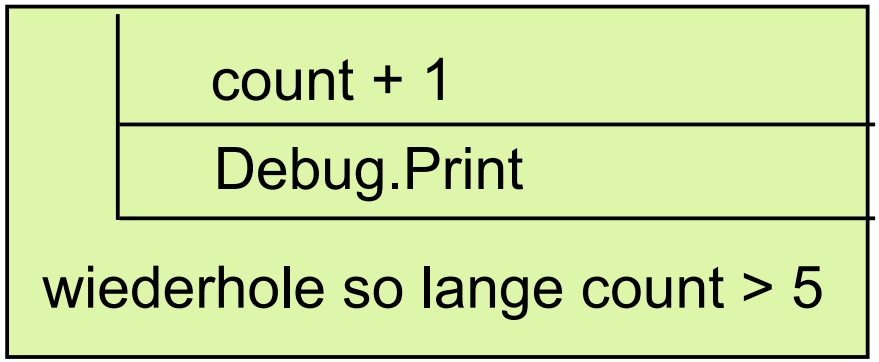
```
count = 0 + 1
count = 1 + 1
count = 2 + 1
count = 3 + 1
count = 4 + 1
count = 5 + 1
    ⇒ Abbruch
```



Do-Loop While

```
Dim count As Integer = 0

Do
    count = count + 1
    Debug.Print count
Loop While count > 5
```

count = 0 + 1
⇒ Abbruch



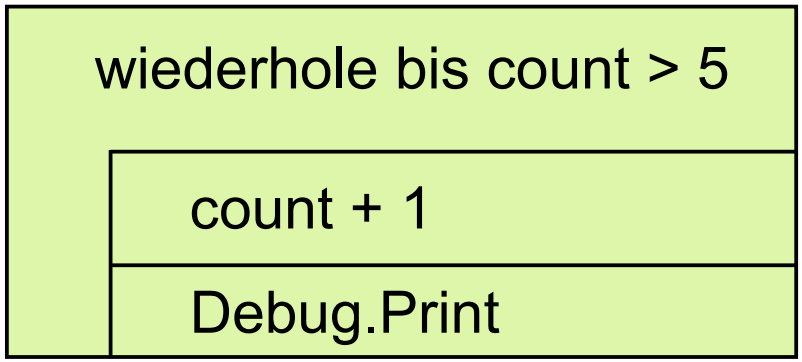
Kopfgesteuerte Schleifen

- Die Abbruchbedingung steht im Kopf der Schleife.
- Die Bedingung wird überprüft. Falls die Bedingung wahr ist, werden die dazugehörigen Anweisungen ausgeführt.
- Bedingung mit Hilfe von Until (bis):
  - Die Schleife läuft solange bis die Bedingung erfüllt ist.
  - Sobald die Bedingung erfüllt ist, wird die Schleife abgebrochen.
- Bedingung mit Hilfe von While (während, so lange):
  - Die Schleife läuft solange wie die Bedingung erfüllt ist.
  - Sobald die Bedingung nicht erfüllt ist, wird die Schleife abgebrochen.

Do-Loop Until

```
Dim count As Integer = 0

Do Until count > 5
    count = count + 1
    Debug.Print count
Loop
```

count = 0 + 1
count = 1 + 1
count = 2 + 1
count = 3 + 1
count = 4 + 1
count = 5 + 1
⇒ Abbruch



Do-Loop While

```
Dim count As Integer = 0

Do While count > 5
    count = count + 1
    Debug.Print count
Loop
```

⇒ Abbruch

wiederhole so lange count > 5

- count + 1
- Debug.Print

Schleife vorzeitig abbrechen

- Exit For verlässt eine For-Schleife und führt alle nachfolgenden Anweisungen aus.
- Exit Do verlässt eine Do-Loop-Schleife und führt alle nachfolgenden Anweisungen aus.