Agenda für heute, 20. April, 2006

- **Wiederholte Programmausführung**
- Algorithmische Grundlagen
- Bedingungen zum Abbruch von Programmschleifen
- Geschachtelte Programmschleifen
- Syntaxdiagramme

Steuerung des Programmablaufs: Wiederholung

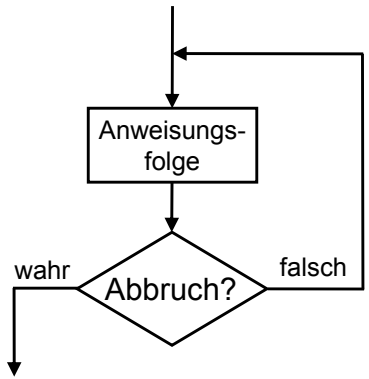
- Gründe für die Wiederholungen von Anweisungsfolgen:
  - Operationen werden solange ausgeführt, bis alle Daten verarbeitet sind
  - eine Berechnung wird mit veränderten Eingabewerten solange wiederholt, bis das Resultat eine gewünschte Genauigkeit erreicht
  - ein Programm soll solange wie gewünscht wiederholt werden
- Ein Programm muss also

  a) wissen was wiederholt ausgeführt werden muss und

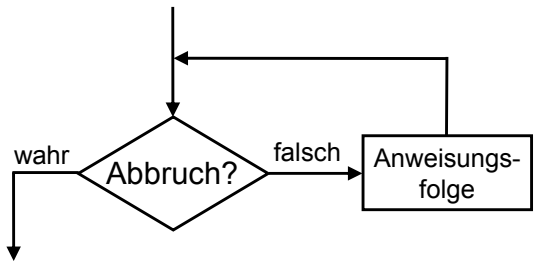
  b) entscheiden können wann anzuhalten ist

- Wiederholte Programmausführung
- **Algorithmische Grundlagen**
- Bedingungen zum Abbruch von Programmschleifen
- Geschachtelte Programmschleifen
- Syntaxdiagramme

Algorithmische Grundlagen: Zwei Arten der Wiederholung



Anweisungsfolge wird mindestens
1 mal ausgeführt



Anweisungsfolge wird nicht,
1 mal oder mehrmals ausgeführt

- Wiederholte Programmausführung
- Algorithmische Grundlagen
- **Bedingungen zum Abbruch von Programmschleifen**
- Geschachtelte Programmschleifen
- Syntaxdiagramme

Wie oft wird eine Schleife ausgeführt?

- Bei einer Wiederholung bewegt sich das Programm in einer Schleife indem es auf eine bereits ausgeführte Instruktion zurück "springt".
- Programmiersprachen stellen für diese Operation eine Anweisung zur Verfügung, die an der Stelle eingefügt wird, an welche die Schleife zurück kehrt.
- In Pascal sind dies die folgenden Schleifen-Anweisungen:

```
for Anfangswert to Endwert do Anweisungsfolge
while Bedingung do Anweisungsfolge
repeat Anweisungsfolge until Bedingung
```

Die for-Schleife führt eine vorgegebene Anzahl Wiederholungen aus.

Die Anzahl Wiederholungen der while- und repeat-Schleife wird durch Aktionen innerhalb der Schleife gesteuert.

Struktur der for-Anweisung

Kontrollvariable (muss deklariert sein)

for *Variable*:= *Ausdruck1* **to** *Ausdruck2* **do** *Anweisung*

Beispiel:

```
for i:= 0 to 9 do write(i)
```

for *Variable*:= *Ausdruck1* **downto** *Ausdruck2* **do** *Anweisung*

Beispiel:

```
for i:= 9 downto 0 do write(i)
```


Struktur der while-Anweisung

while *boolescher Ausdruck* **do** *Anweisung*

Die Anweisung wird wiederholt solange ausgeführt wie eine Auswertung des booleschen Ausdrucks den Wahrheitswert "true" ergibt.

Beispiel:

```
i:= 0;
while i < 10 do
begin
   write(i);
   i:= i + 1
end;
```

Struktur der repeat-Anweisung

repeat *Anweisung* **until** *boolescher Ausdruck*

Führe Anweisung einmal aus und wiederhole solange bis eine Auswertung des booleschen Ausdrucks den Wahrheitswert "true" ergibt.

Beispiel:

```
i:= 0;
repeat write(i); i:= i + 1 until i = 10
```


Schachtelung am Beispiel der for-Anweisung

Die Anweisung innerhalb einer Schleife kann beliebige Anweisungen enthalten, insbesondere auch **Wiederholungs-Anweisungen**.

for *Variable1*:= *Ausdruck1* **to** *Ausdruck2* **do**
  for *Variable2*:= *Ausdruck3* **downto** *Ausdruck4* **do** *Anweisung*

Beispiele:

```
for i:= 1 to 3 do
  for j:= 1 to 10 do write(i*j)

for i:= 1 to 3 do
  begin
    for j:= 1 to 10 do write(i*j);
    writeln;
  end;
```

Grundlagen der Syntaxdiagramme

Elemente der Syntaxdiagramme

Nicht-Terminalsymbole: Bezeichner

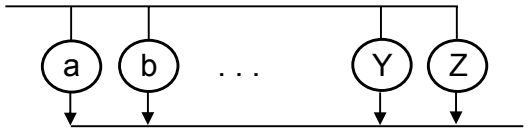
Terminalsymbole: ;

Nicht-Terminalsymbole können durch andere Nicht-Terminalsymbole oder durch Terminalsymbole nach vorgegebenen Syntaxregeln ersetzt werden.

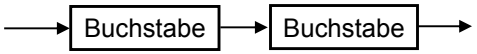
Grundlagen der Syntaxdiagramme

Produktionsregeln (gelten für beide Arten von Symbolen)

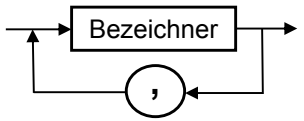
Auswahl:



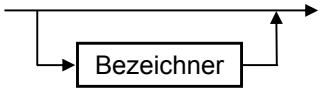
Folge:



Wiederholung:

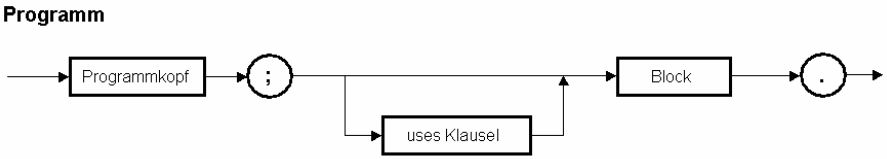


Option:

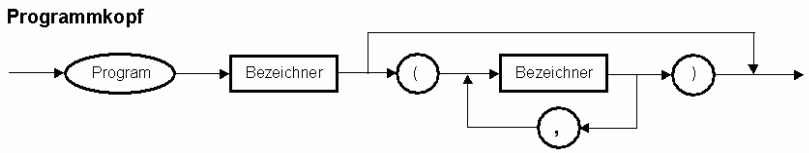


Syntaxregeln am Beispiel "Programm"

Programm

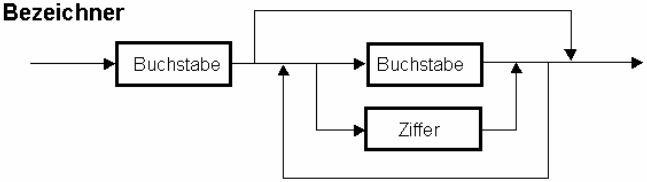


Programmkopf



Syntaxregeln am Beispiel "Bezeichner"

Bezeichner



Ziffer