2 Grundlegende Kontroll- und Datenstrukturen

2.1 Kontrollstrukturen

Jede algorithmische Problemlösung lässt sich durch nur drei grundlegende Kontrollstrukturen (Sequenz, Auswahl und Wiederholung) und dem Problem angepasste Daten beschreiben. Die Übersetzung einfacher Algorithmen in eine Programmiersprache zeigt, dass auch komplexe Programme in einer sehr einfachen und überschaubaren Sprache formuliert sind.

Sequenz

Selten kann ein Problem durch eine einzige Anweisung gelöst werden. Meist ist eine Anzahl von Anweisungen nötig. Im einfachsten Fall werden diese in bestimmter Reihenfolge ein einziges Mal abgearbeitet.

> Definition: Eine Folge von Anweisungen, die alle nacheinander in bestimmter Reihenfolge nur einmal abgearbeitet werden, heißt Sequenz.

Beispiel 1: Berechne zu einem Kapital K bei einem Zinssatz p (in %) für die Zeit t (in Tagen) die Zinsen Z.
Mathematische Grundlagen:
$Z = \frac{K \cdot p \cdot t}{100 \cdot 360}$

Struktogramm:

Gib K, p und t ein.
Berechne Z durch Z = (K*p*t)/(100*360)
Schreibe den Wert von Z auf den Schirm.
Programmende.

Bei der Codierung soll gleich auf einen sinnvollen Dialog zwischen Mensch und Maschine geachtet werden, damit auch ein ungeübter Benutzer das Programm bedienen kann.

Codierung in Turbo-Pascal:

```
program programm2a_(zinsen);

uses crt;
var  k, p, z:  real;
     t:        integer;

begin
     clrscr;   {löscht den Bildschirm}
     writeln('Zinsberechnung');
     writeln;  {Leerzeile}
     write('Kapital in DM: ');readln(k);
     write('Zinssatz in %: '); readln(p);
     write('Zeit in Tagen: '); readln(t);
     z := (k*p*t)/(100*360);  {Zinsberechnung}
     writeln;
     writeln('Zinsen in DM: ', z:10:2);
     readln
end.
```

Anmerkungen:

1. Die **WRITELN**-Anweisungen allein bewirken Zeilenvorschübe zur optischen Gliederung der Bildschirmausgabe.
2. Mit der Wertzuweisung durch den Zuweisungsoperator "**:=**" wird einer Variablen ein Wert zugeordnet. Diese Variable steht links vom Zuweisungsoperator, rechts steht der Term, dessen Wert zugeordnet werden soll.
3. Im Beispiel werden zwei Datentypen verwendet: reelle (**real**) und ganze (**integer**) Zahlen.

Beispiel 2: Erfrage sukzessive eine Adresse und gib sie in Form eines Adressaufklebers auf den Drucker aus.

Struktogramm:

Gib Name, Vorname, Straße, PLZ und Wohnort ein.
Schreibe (auf dem Drucker) "Herrn/Frau/Frl./Firma".
Schreibe Vorname und Nachname.
Schreibe die Straßenbezeichnung.
Schreibe nach einer Leerzeile PLZ und Wohnort.

Anmerkung:
Vor der Codierung soll gleich auf den Variablentyp "**STRING**" eingegangen werden. Unter einem String versteht der Computer eine Zeichenkette, deren maximal erwartete Länge in eckigen Klammern hinter **STRING** angegeben wird. Fehlt die Angabe, nimmt Turbo-Pascal eine maximale Stringlänge von 255 an. Stellen im String nach der maximal vereinbarten Stringlänge werden ignoriert.

Codierung in Turbo-Pascal:

```
program programm3_(adresse);

uses crt, printer;
var  name, vorname, strasse, wohnort: string[20];

begin
     clrscr;
     writeln('Adressenausdruck');
     writeln;
     write('Name: ');          {Beginn Eingabeteil}
     readln(name);
     write('Vorname:'); readln(vorname);
     write('Straße/Nr.:'); readln(strasse);
     write('PLZ/Wohnort:'); readln(wohnort);
     writeln(lst, 'Herrn/Frau/Frl.'); {Beginn Ausgabeteil}
     write(lst, vorname, ' ');{Leerzeichen nach vorname}
writeln(lst, name);
     writeln(lst, strasse);
     writeln(lst);
     writeln(lst, wohnort)
end.
```

Anmerkung: Der Ausdruck auf dem Drucker erfordert einerseits die Anweisung **USES PRINTER**, andererseits muss nach einem **WRITE**-Befehl dem Drucker durch **WRITE***(lst, ...)* mitgeteilt werden, dass eine Umleitung der Ausgabe auf den Drucker erfolgen soll.

Ein- und zweiseitige Auswahl

Häufig muss zur Lösung eines Problems an einer oder mehreren Stellen eine Entscheidung darüber getroffen werden, wie weiter zu verfahren ist. Derartige Entscheidungen kann der Computer mit der Anweisung "Wenn - dann - sonst" treffen.

In Pascal dienen die Schlüsselwörter **IF - THEN - ELSE** zur Formulierung einer zweiseitigen Auswahl.

Anmerkungen hierzu:

1. Die zweiseitige Auswahl hat das Format **IF** *Bedingung* **THEN** *Anweisung* **ELSE** *Anweisung*.
2. Eine zweiseitige Auswahl besteht aus dem Wenn-Teil (er enthält die Prüfbedingung), dem Dann-Teil (er enthält die Anweisungen, die ausgeführt werden, wenn die Bedingung erfüllt ist), und evtl. dem Sonst-Teil (er enthält die Alternativanweisungen).
3. Zur Formulierung von Bedingungen können neben mathematischen auch logische Operatoren stehen.
   Beispiel für Bedingungen: **IF** (a > 2 **AND** a < 5) **THEN** ...
4. Die **IF-THEN-ELSE**-Anweisung bildet einen einzigen, aus mehreren Teilen zusammengesetzten Satz. Folgen z. B. hinter **THEN** mehrere Anweisungen, so müssen diese der Eindeutigkeit wegen durch die Wörter **BEGIN** und **END** geklammert werden.
5. Vor **ELSE** darf kein Strichpunkt stehen.

Beispiel: Ermitteln Sie die Lösungen einer beliebigen quadratischen Gleichung der Form $a \cdot x^2 + b \cdot x + c = 0$.

Mathematische Grundlagen:
Die Lösungen der quadratischen Gleichung sind gegeben durch
$x_{1/2} = \frac{-b \pm \sqrt{D}}{2 \cdot a}$ mit $D = b^2 - 4 \cdot a \cdot c$,
falls die Diskriminante $D \geq 0$ (und $a \neq 0$) ist.

Kurzbeschreibung des Algorithmus:
Gib die Koeffizienten a, b und c ein. Setze D := b^2 - 4 · a · c. Wenn D < 0 ist, dann schreibe "keine reelle Lösung!", sonst schreibe die Lösungen hin.

Das Struktogramm hat folgendes Aussehen:

<table>
<tr><td colspan="2">Lösche den Bildschirm.</td></tr>
<tr><td colspan="2">Schreibe "Lösung einer quadratischen Gleichung"</td></tr>
<tr><td colspan="2">Lies den ersten Koeffizienten a ein. (CR)</td></tr>
<tr><td colspan="2">Lies den zweiten Koeffizienten b ein (CR)</td></tr>
<tr><td colspan="2">Lies den dritten Koeffizienten c ein. (CR)</td></tr>
<tr><td colspan="2">Weise der Variablen D den Wert der Diskriminanten zu.</td></tr>
<tr><td>ja
Ist D < 0?</td><td>nein</td></tr>
<tr><td rowspan="2">Schreibe "Keine reelle Lösung!"</td><td>Schreibe "x1 = ", (-b + sqrt(D))/(2 * a)</td></tr>
<tr><td>Schreibe "x2 = ", (-b - sqrt(D))/(2 * a)</td></tr>
</table>

Anmerkung:
Selbstverständlich können beim Einlesen der Werte und bei der Ausgabe der Lösungen erläuternde Texte hinzugefügt werden!

In Pascal kann dieser Algorithmus so codiert werden:

```
program programm4_(quadratische_gleichung);

uses crt;
var       a, b, c, d: real;

begin     {Eingabe}
     clrscr;   {löscht den Bildschirm}
     writeln('Lösung der quadratischen Gleichung');
     writeln('a*x*x + b*x + c = 0');
     writeln;
     write('1. Koeffizient: '); readln(a);
     writeln;
     write('2. Koeffizient: '); readln(b);
     writeln;
     write('3. Koeffizient: '); readln(c);
     writeln; writeln;
     {Lösung}
     d := sqr(b) - 4*a*c;     {Diskriminante}
     if d < 0            {Bedingung }
       then writeln('Keine reelle Lösung!')  {Dann-Teil}
       else
          begin                              {Sonst-Teil}
            writeln('1. Lösung: x1 = ',(-b + sqrt(d))/(2*a));
            writeln;
            writeln('2. Lösung: x2 = ',(-b - sqrt(d))/(2*a))
          end;
     readln
end.
```

Mehrfachauswahl

Neben ein- und zweiseitiger Auswahl ist auch mehrseitige Auswahl möglich, bei der eine von mehreren Möglichkeiten ausgewählt wird (z. B. in Auswahlmenüs!). In Pascal dienen die Schlüsselwörter **CASE OF - ELSE - END** zur Auswahl einer Möglichkeit bei einer mehrfachen Fallunterscheidung.

Anmerkungen zur Realisierung in Pascal:

1. Diese Kontrollanweisung hat das Format
   CASE *(beliebiger Text)* **OF** *(beliebiger Wert 1): Anweisung 1; (beliebiger Wert 2): Anweisung 2; ...* **ELSE** *(Anweisung)* **END**;
2. Welcher Fall ausgewählt wird, hängt vom Wert des sog. Selektors (zwischen **CASE** und **OF**) ab.

3. Die Fälle werden durch eine Liste von Werten (keine Variable oder Terme!) gekennzeichnet. Realzahlen dürfen nicht verwendet werden.
4. Mehrere Anweisungen in einem Fall sind durch **BEGIN** und **END** zu klammern.
5. In Turbo-Pascal kann nach der Aufzählung der möglichen Fälle nach **ELSE** ein Anweisungsteil folgen, der ausgeführt wird, wenn keiner der vorher beschriebenen Fälle zutrifft.
6. Die ganze Fallunterscheidung wird mit **END** abgeschlossen; vor dem End braucht kein Semikolon zu stehen.

Beispiel: Nach Eingabe einer (sinnvollen) Note soll das Programm folgende Kommentare ausgeben: bei Note 1 oder 2 "gefällt mir!", bei Note 3 "Mittelmaß; strenge dich mehr an!" und bei 4, 5 oder 6 "Leider die schlechtere Hälfte!". Wird eine andere natürliche Zahl eingegeben, dann soll die Meldung "Eingabefehler!" erscheinen.

Das Struktogramm hat folgendes Aussehen:

<table>
<tr><td colspan="4">Schreibe "Mehrfachauswahl"</td></tr>
<tr><td colspan="4">Lies eine Note ein und weise sie der Variablen NOTE zu.</td></tr>
<tr><td colspan="4">Falls NOTE =
1,2 | 3 | 4,5,6 | sonst</td></tr>
<tr><td>Schreibe: "Gefällt mir!"</td><td>Schreibe: "Mittelmaß! Strenge dich mehr an!"</td><td>Schreibe: "Leider die schlechtere Hälfte!"</td><td>Schreibe: "Eingabe-fehler!"</td></tr>
<tr><td colspan="4">Schreibe: "Ende des Programms Mehrfachauswahl"</td></tr>
</table>

Die Codierung in Pascal kann so aussehen:

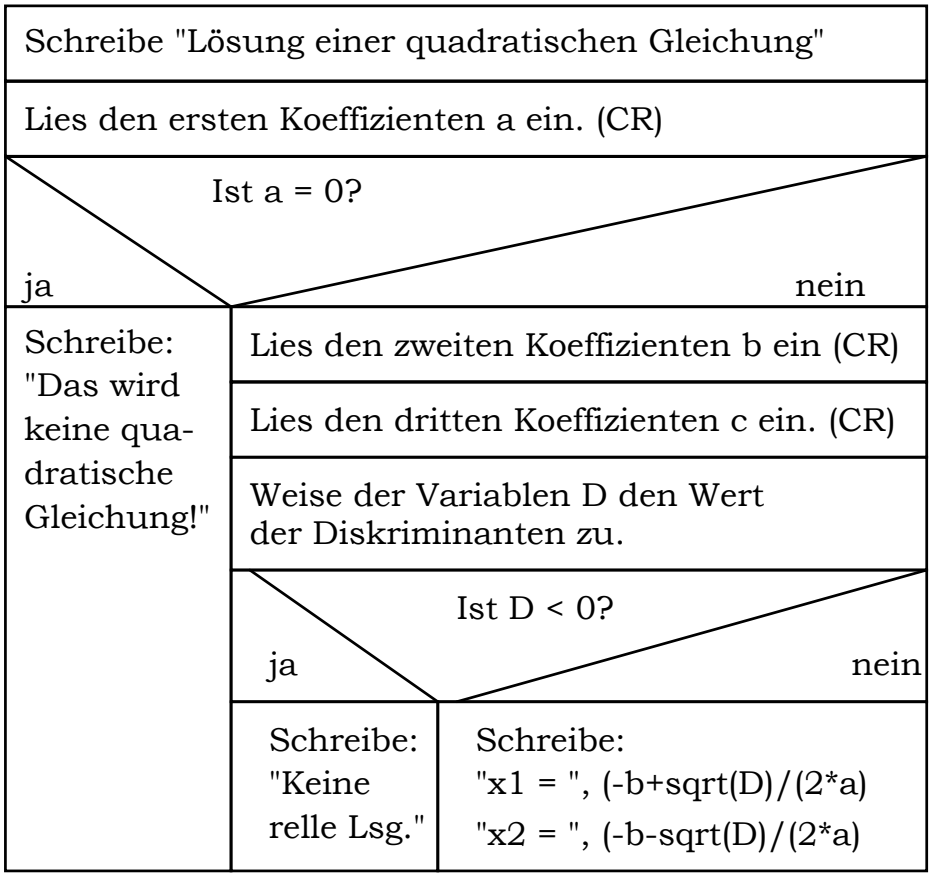
```
program programm5_(mehrfachauswahl);

uses crt;
var     note: integer;

begin
     clrscr; (*löscht den Bildschirm*)
     writeln('Mehrfachauswahl'); writeln;
     write('Welche Note möchtest du eingeben? ');
     readln(note); writeln;
```

```
     case note of
          1,2:      writeln('Gefällt mir!');
          3:        begin
                         writeln('Mittelmaß!');
                         writeln('Strenge dich mehr an!')
                    end;
          4,5,6:    writeln('Leider die schlechtere Hälfte!')
     else writeln('Eingabefehler!')
     end;
     writeln;
     writeln('Ende des Programms');
     readln
end.
```

Häufig ist eine Auswahl von einer vorausgehenden Auswahl abhängig. Eine solche Auswahl heißt dann mehrstufig oder geschachtelt.

Beispiel: Löse einmal mehr die quadratische Gleichung $a \cdot x^2 + b \cdot x + c = 0$; bedenke dabei, dass für a = 0 keine quadratische Gleichung vorliegt und der eingesetzte Lösungsalgorithmus versagt.

Das Struktogramm hat folgendes Aussehen:



Anmerkung:

Bei geschachtelten Anweisungen ist eine übersichtliche Schreibweise besonders wichtig. Sie wird erreicht

1. durch Einrückungen zum Anzeigen der Schachtelungstiefe
2. durch untereinander schreiben entsprechender Schlüsselwörter
3. durch Einfügen von Kommentaren.

Das zugehörige Pascal-Programm kann so aussehen:

```
program programm6_(geschachtelte_auswahl);

uses crt;
var  a, b, c, d: real;

begin               (* Eingabe *)
  clrscr;                (*löscht den Bildschirm*)
    writeln('Lösung der quadratischen Gleichung');
    writeln('a*x*x + b*x + c = 0');
    writeln;
    write('1. Koeffizient: '); readln(a);
    writeln;
  if a = 0
  then writeln('Das wird keine quadratische Gleichung!')
         (* 1. Stufe der Auswahl *)
  else
    begin
      write('2. Koeffizient: '); readln(b);
      writeln;
      write('3. Koeffizient: '); readln(c);
      writeln; writeln;       (* Lösung *)
      d := sqr(b) - 4*a*c;    (* Diskriminante *)
      if d < 0                (*Bedingung *)
      then writeln('Keine reelle Lösung!')(*Dann-Teil *)
      else
        begin
          writeln('1. Lösung: x1 = ',(-b+sqrt(d))/(2*a):5:3);
          writeln;
          writeln('2. Lösung: x2 = ',(-b - sqrt(d))/(2*a):5:3)
        end
    end;
    readln
end.
```

Anmerkung:
Durch dieses Programm sind nicht alle möglichen Fälle erfasst. Ein vollständiges Programm zur Lösung der quadratischen Gleichung müsste zum Beispiel auch die nachfolgenden Fälle erfassen:

1. jede reelle Zahl ist Lösung (für a = 0, b = 0, c = 0)
2. die Gleichung hat eine einfache Lösung (a = 0, b ≠ 0)
3. die Gleichung hat eine Doppellösung (a ≠ 0, D = 0).

Wiederholung mit Zähler

Bei vielen Algorithmen sind Anweisungen wiederholt auszuführen, wobei die Anzahl der Wiederholungen von verschiedenen Bedingungen abhängen kann. So kann etwa die Zahl der Wiederholungen von vornherein festgelegt sein (Beispiel: Einmaleins) oder von Prüfbedingungen innerhalb des Algorithmus z. B. am Anfang oder am Ende abhängen (bedingte Wiederholung).

Beispiel für Wiederholung mit Zähler: Es soll die Entwicklung eines Anfangskapitals K_0 im Verlauf von 10 Jahren bei jährlicher Verzinsung mit dem Zinsfuß p berechnet werden, wenn die jährlichen Zinsen jeweils dem Kapital zugeschlagen werden (Beispiel: Sparbriefe, Kommunalobligationen, Bundesanleihen).

Mathematische Überlegungen: Die jährlichen Zinsen Z errechnen sich aus dem Kapital K am Jahresanfang nach der einfachen Gleichung
$Z = K \cdot \frac{p}{100}$.
Nach jedem Jahr ist das im kommenden Jahr zu verzinsende Kapital zu aktualisieren:
$N_{eu} = K_{alt} + Z$
Diese Erhöhung des zu verzinsenden Kapitals ergibt den "Zinseszins-Effekt".

Das Struktogramm für den Algorithmus sieht so aus:

Schreibe "Zinseszins"
Lies Kapital k, Zinssatz p und Laufzeit j ein.
Schreibe "Jahr", Kapital".
Wiederhole für Jahr von 0 bis j
Setze z := k*p/100
Setze k := k + z
Schreibe Jahr, k
Schreibe: "Programmende"

Anmerkungen zur Codierung in Pascal:

1. Die Wiederholung mit vorgegebener fester Abbruchbedingung wird realisiert durch die **FR - DO** - Schleifenanweisung. Mit ihr wird eine Anweisung oder eine Sequenz so oft wiederholt, bis die Zählervariable

alle Werte von einem Anfangswert bis zu einem Endwert mit der Schrittweite 1 durchlaufen hat. Die Syntax dazu lautet:
FR *Zählervariable := Anfangswert* **TO/DOWNTO** *Endwert* **DO** *Anweisung(en)*

2. Soll abwärts gezählt werden, muss das Schlüsselwort **DO** durch **DOWNTO** ersetzt werden.
3. Besteht der Anweisungsblock aus mehr als einer Anweisung, dann ist mit **BEGIN** und **END** zu klammern.
4. Zur Angabe von Anfangs- und Endwert der Zählervariablen dürfen auch Terme verwendet werden (z. B. Zählervariable := x + 2).
5. Die Zählervariable und die Anfangs- und Endwerte müssen vom gleichen abzählbaren Typ sein.
6. Der Wert der Zählervariablen darf im Anweisungsblock nicht verändert werden.

Codierung in Pascal:

```
program programm7_(wiederholung_mit_zaehler);
uses crt;
var       k, p, z:  real;
     jahr, j:  integer;

begin
     clrscr; {löscht den Bildschirm}
     writeln('Entwicklung eines Kapitals K bei p Prozent');
     writeln('in j Jahren (Zinseszinseffekt)');
     writeln; writeln;
     write('Kapital:  ');     readln(k);
     writeln;
     write('Zinssatz: ');     readln(p);
     writeln;
     write('Laufzeit: ');readln(j);
     writeln; writeln;
     write('Jahr ','Kapital');
     writeln; writeln;
     for jahr := 1 to j do
          begin
               z := k * p / 100;
               k := k + z;
               write(jahr,'    ',k);
               writeln
          end;
     writeln;
     writeln('Programmende');
     readln
end.
```

Anmerkung zum konkreten Beispiel:

1. Eine analytische Betrachtung des Problems zeigt, dass das Endkapital nach j Jahren mit der Gleichung $K = K_0 \cdot (1 + \frac{p}{100})^j$

direkt berechnet werden kann, ohne Zwischenwerte berechnen zu müssen.

2. Bei einigen Anlageangeboten wird anstelle des aussagekräftigen Begriffs "Rendite" (hier identisch mit dem Nominalzinssatz) der Begriff des "Wertzuwachses" verwendet. Dabei wird ohne Berücksichtigung des Zinseszinseffekts ein mittlerer jährlicher Zins nach der Gleichung
$\text{Wertzuwachs} = \frac{K - K_0}{K_0 \cdot j} \cdot 100\%$
berechnet. Dieser Wert ist höher als der effektive Zins, aber zum Renditenvergleich ungeeignet, da er den vor allem bei langen Laufzeiten wesentlichen Zinseszinseffekt nicht berücksichtigt!
Beispiel: Bei einem Kapital K_0 = 1000 DM ergibt sich bei einem Zinssatz p = 5 (= effektiver Zins) in j = 20 Jahren 2653,29 DM. Dagegen liefert die Gleichung für den Wertzuwachs stolze 8.2 %!
3. Anleihen sind zwar mit festem Zins für die gesamte Laufzeit ausgestattet, im Gegensatz zu den Sparpapieren können Anleihen aber jederzeit vor Fälligkeit wieder über die Börse verkauft werden. Der Preis wird von Angebot und Nachfrage bestimmt. So werden in Zeiten steigenden Zinses bereits laufende Anleihen im Kurs sinken, um die niedrigere Nominalverzinsung auszugleichen und eine marktgerechte Verzinsung zu bieten.
Zur Berechnung der Rendite müssen Nominalzins p, (restliche) Laufzeit j und Börsenkurs BK bekannt sein. Dann lässt sich mit der nachfolgenden Gleichung die Rendite näherungsweise berechnen:
$\text{Rendite} = \left(p + \frac{100 - BK}{j}\right) \cdot \frac{100}{BK}$.
Beispiel: p = 6, j = 2, BK = 96 => Rendite = 8,33

Bedingte Wiederholung

Nicht immer steht die Anzahl der Wiederholungen von vornherein fest. Es ist vielmehr während des Programmlaufes zu entscheiden, ob noch einmal wiederholt werden soll oder nicht.

Bei dieser "bedingten Wiederholung" ist zu unterscheiden, ob zu wiederholen ist, bis ein bestimmtes Ziel erreicht ist (Wiederholung mit Endbedingung) oder ob zu wiederholen ist, solange bestimmt Umstände dies erfordern (Wiederholung mit Anfangsbedingung).

1. Beispiel (Wiederholung mit Abbruchbedingung am Ende): Berechne die kleinste Quadratzahl, die größer ist als eine frei zu wählende Zahl.

Anmerkungen zur Codierung in Pascal:

1. Die Wiederholungsanweisung mit Endbedingung wird mit der Schleifen-Kontrollanweisung **REPEAT - UNTIL** (wiederhole - bis) formuliert. Sie hat die Syntax
REPEAT *Anweisungen* **UNTIL** *Bedingungen*;

2. Die beiden Schlüsselwörter **REPEAT** und **UNTIL** bilden bereits eine Wortklammer um den Anweisungsteil; eine zusätzliche Klammerung mit **BEGIN - END** ist daher nicht erforderlich!
3. Der Anweisungsteil wird mindestens 1-mal ausgeführt.

Struktogramm:

<table>
<tr><td colspan="2">Schreibe "Kleinste Quadratzahl über einer festen Zahl"</td></tr>
<tr><td colspan="2">Lies die Schwellenzahl z ein.</td></tr>
<tr><td colspan="2">Setze x := 0</td></tr>
<tr><td rowspan="3">Wiederhole</td><td>Setze x := x + 1 {Zähler erhöhen}</td></tr>
<tr><td>Setze y := x * x</td></tr>
<tr><td colspan="2">bis y > z</td></tr>
<tr><td colspan="2">Schreibe "Die kleinste Quadratzahl über ", z, " heißt ", y</td></tr>
</table>

Codierung in Pascal:

```
program programm8_(wiederholung_mit_Bedingung_am_Ende);
uses crt;
var       x, y, z: integer;

begin
     clrscr;
     writeln ('Kleinste Quadratzahl über einer festen Zahl');
     writeln; writeln;
     write ('Zahl: '); readln (z);
     writeln;
     x := 0;
     repeat
          x := x + 1; { Zähler um 1 erhöhen}
          y := x * x
     until y > z;
     writeln ('Die kleinste Quadratzahl, die größer ist');
     writeln ('als ', z:5, ' heißt ', y:5, ';');
     writeln ('sie ist das Quadrat von ', x:5, '.');
     writeln;
     writeln ('Programmende');
     readln
end.
```

2. Beispiel (Wiederholung mit Abbruchbedingung am Anfang): Berechne die Quadratwurzel einer positiven Zahl z durch fortgesetzte Intervallhalbierung.

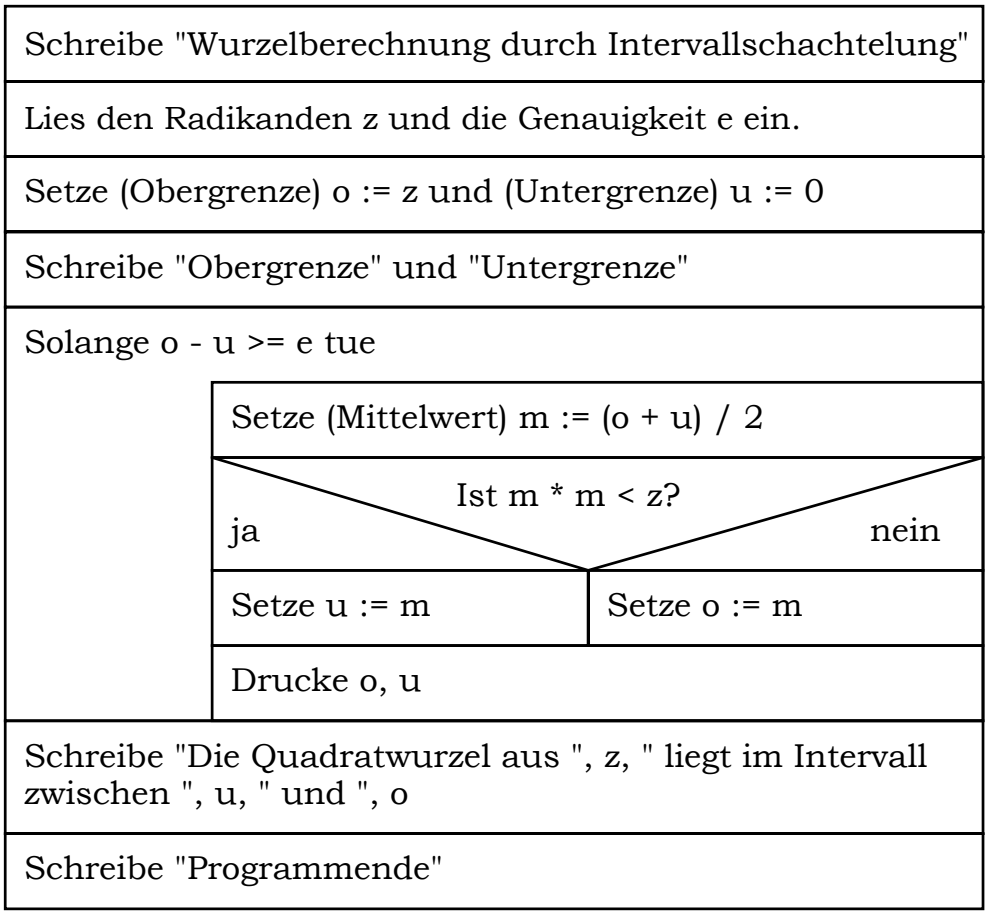
Mathematische und strategische Überlegungen:

Nach Festlegung von Ober- und Untergrenzen (max. 0 und z) wird das Intervall halbiert. Dann wird mit einer Abfrage (ist (Intervallmitte)^2 < z?) untersucht, ob die Intervallmitte zu groß oder zu klein ist; entsprechend dem Ergebnis dieser Prüfung ist diese Intervallmitte die neue Ober- bzw. Untergrenze des neuen Intervalls.

Auf diese Weise wird die Intervalllänge durch fortgesetzte Intervallschachtelung immer kleiner und jedes Intervall liegt in allen vorangegangenen Intervallen; dies ist die Bedingung für die Darstellung einer reellen Zahl durch eine Intervallschachtelung.

Die Intervallschachtelung wird nicht mehr weitergeführt, wenn sich Ober- und Untergrenze des Intervalls um weniger als ein vorher festgelegtes Maß unterscheiden.

Das Struktogramm kann so aussehen:



Anmerkungen zur Codierung in Pascal:

1. Die Wiederholungsanweisung mit Anfangsbedingung wird mit den Schlüsselwörtern **WHILE - DO** gebildet. Sie hat die Syntax **WHILE** *Bedingungen* **DO** *Anweisung(en)*
2. Stehen nach **DO** mehrere Anweisungen, dann ist in üblicher Weise mit **BEGIN - END** zu klammern.
3. Der Anweisungsteil wird unter Umständen überhaupt nicht ausgeführt.

Das Pascal-Programm kann so aussehen:

```
program programm9_(wiederholung_mit_Bedingung_am_Anfang);
uses crt;
var  o, u, m, z, e: real;

begin
     clrscr;
     writeln ('Wurzelberechnung durch fortgesetzte
Intervall-');
     writeln ('schachtelung und Abbruchbedingung am Anfang');
     writeln; writeln;
     write ('Zahl (> 0)                : '); readln (z);
     write ('Genauigkeit (z. B. 0.001): '); readln (e);
     writeln; writeln;
     o := z; u := 0;
     write ('Obergrenze ', 'Untergrenze');
     writeln; writeln;
     while (o - u >= e) do
          begin
               m := (o + u) / 2;
               if m * m < z
                    then u := m
                    else o := m;
               write (o:5:5, '    ', u:5:5);
               writeln
          end;
     writeln ('Die Quadratwurzel aus ', z:5:5);
     writeln ('liegt im Intervall zwischen');
     writeln;
     writeln (u:5:5, ' und ', o:5:5);
     writeln; writeln;
     writeln ('Programmende');
     readln
end.
```

Am Schluss dieses Kapitels sei nochmals auf die Unterschiede zwischen den verschiedenen Schleifenarten hingewiesen:

Die **FR - DO** - Schleife wird sinnvollerweise verwendet, wenn die Zahl der Schleifendurchläufe bekannt und während des Programmlaufs nicht mehr verändert werden muss. Die Schrittweite ist immer 1.

Bei der **REPEAT - UNTIL** - Schleife wird die Bedingung nach Ausführung der Schleifenanweisung auf den Wahrheitswert **TRUE** überprüft, bei der **WHILE - UNTIL** - Schleife vorher. Die **REPEAT**-Schleife läuft, bis eine Bedingung erfüllt ist, die **WHILE**-Schleife, solange eine Bedingung erfüllt ist.