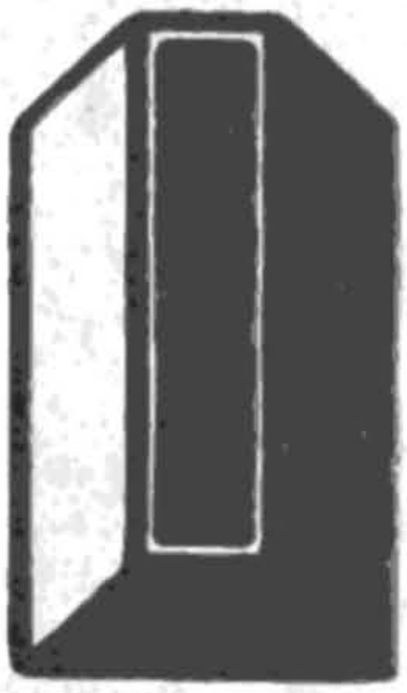
# Kleinstrechner

Tendenzen
und Theorien

Verstärkerstufen



Informationen
und Ideen

Kalenderfunktionen

Programme
und Projekte

Eingaberoutinen

Spaß
und Spiel

6

# Kleinstrechner-TIPS

**Heft 6**

Mit 25 Bildern

Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Doz. Dr.-Ing. Wilhelm Leupold
Doz. Dr. sc. techn. Thomas Horn

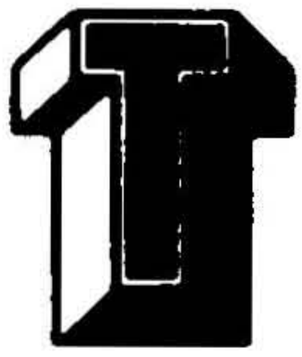
**VEB Fachbuchverlag Leipzig**

# Inhalt



ISBN 3-343-00226-7

# Zum Inhalt



Auch im Heft 6 der »Kleinstrechner-TIPS« werden einem breiten Interessentenkreis Anregungen für den Computereinsatz gegeben sowie Beiträge zur Erweiterung des Grundwissens der Informatik und Beispielprogramme – vorzugsweise in BASIC – veröffentlicht.

SCHILLING führt in die formalisierte Beschreibung von Algorithmen durch Verwendung eines Pseudocodes ein. Gezeigt wird diese Entwurfsmethodik am Beispiel eines »Rechentrainers« für Schüler der 9. und 10. Klasse der Polytechnischen Oberschule.

MICHEL geht in seinem Beitrag »Wieso rechnet (m)ein Taschenrechner $|\bar{8} \times |\bar{8} = 8$?« dem Problem der Darstellungsgenauigkeit von irrationalen Zahlen auf rechentechnischen Hilfsmitteln nach.

Bei vielen Problematiken werden Zufallszahlen benötigt. Im Teil 1 einer Folge von Beiträgen zur Simulation auf Mikrorechnern erklären LORENZ und SCHULZE den Begriff der Zufallszahl, zeigen die Möglichkeiten ihrer Generierung und geben dafür Programme an.

In mehreren Beiträgen von KÜHNEL und HÄHNEL werden dem Leser BASIC-Programme für ein Nachvollziehen oder für das Variieren und Einbauen in eigene Programme vorgestellt. Kalenderfunktionen werden oft z.B. im Zusammenhang mit der Archivierung von Informationen benötigt. Ein anderes BASIC-Programm enthält eine „intelligente" Eingaberoutine für komplexe Zahlen, bei der keine bestimmte Darstellungsform der einzugebenden Zahl vorgeschrieben ist. Wie einfache Verstärkerstufen berechnet werden können, zeigt ein weiteres BASIC-Programm. Der Artikel verzichtet dabei bewußt auf schaltungstechnische Details.

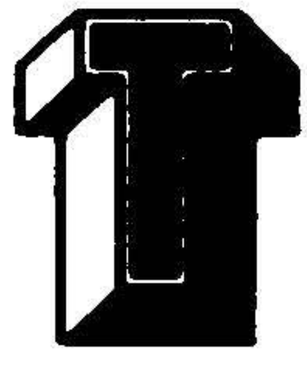
Von SCHÖNFELDER wird ein Programm für ein Master-Mind-Zahlenratespiel beschrieben, das auf dem im Heft 5 vorgestellten Betriebssystem eines selbstgebauten Heimcomputers lauffähig ist.

Auch der Beitrag von KREUL führt in das Reich der Zahlenratespiele. Besonderes Interesse dürfte er dadurch wecken, daß das entwickelte Programm in der weit verbreiteten Programmiersprache PASCAL geschrieben und getestet wurde.

Herausgeber und Verlag hoffen, auch mit diesem Heft den Lesern Unterstützung und Hinweise für ein selbständiges Arbeiten am Computer zu geben. Danken möchten wir auch allen Lesern, die uns Meinungsäußerungen zu den bisher erschienenen Heften und Vorschläge zu ihrer Gestaltung übermittelten.

*Wilhelm Leupold*

# Ein Formalismus zur Beschreibung von Problemlösungen



## Einführung

Als Nutzer eines Kleincomputers sieht man sich bald vor die Notwendigkeit gestellt, eigene Programme zu erarbeiten, da für die zu lösenden Probleme keine Programme käuflich zu erwerben sind bzw. man das Geld für den Kauf nicht zur Verfügung hat. Bei kleineren Aufgabenstellungen besteht das Problem im allgemeinen nur im Erlernen und Beherrschen der Programmiersprache. Hat man diese Klippe erfolgreich gemeistert und traut sich an größere Aufgaben, so sind bis zur Problemlösung doch genauere Überlegungen und meistens mehrere Lösungsschritte notwendig. Es hat sich als günstig erwiesen, das Problem systematisch und schrittweise zu lösen und für die Darstellung der einzelnen Teillösungen eine übersichtliche und damit gut lesbare Form zu verwenden. Für eine solche Lösungsbeschreibung haben sich in der Informatik im wesentlichen zwei Möglichkeiten herausgebildet:

a) graphische Beschreibung in Form von Programmablaufplan oder Struktogramm,

b) verbale Beschreibung in einer Kurzform, z. B. in einer Art Pseudocode.

Jede dieser Darstellungsformen hat ihre Vor- und Nachteile, die meistens auch noch individuell gewichtet werden. Einige dieser Vor- und Nachteile sollen im folgenden aufgeführt werden.
Als Vorteil der graphischen Darstellung wird die zweidimensionale Abbildung der Problemlösung gesehen. Aber gerade darin liegt auch der Nachteil, denn größere Probleme lassen sich dann nur auf mehreren Blättern darstellen, und zum anderen ist immer ein Umdenken notwendig, wenn man, ausgehend von der Lösungsbeschreibung, auf die graphische Darstellung übergeht. Ein weiterer Nachteil der Diagrammformen ist darin zu sehen, daß sie keine Beschreibung von Datentypen und Datenstrukturen ermöglichen, die doch wesentliche Bestandteile jeder Problemlösung sind.. Ebenso sind Vereinbarungs- und Parameterübergabemechanismen keine Bestandteile von Ablaufdiagrammen.
Die verbale formalisierte Beschreibung der Problemlösung vermeidet eine ganze Reihe der Nachteile der graphischen Beschreibung, da eine strukturierte Darstellung in gut lesbarer und sich selbst dokumentierender Form möglich ist. Die Vereinbarung von Datentypen und Datenstrukturen ist ebenso möglich wie die Darstellung der Parameterübergabemechanismen. Auch ist der Aufwand für einen Lösungsschritt wesentlich geringer, da keinerlei Zeichnungen anzufertigen sind. Für den schon länger auf dem Gebiet der

Informatik Tätigen bedeutet die Erstellung der Problemlösung in einer Art Pseudocode eine wesentliche Umstellung gewohnter Arbeitsweisen, was von ihm als Nachteil empfunden wird. Der Informatikneuling wird hier keine Probleme haben und sich schnell einarbeiten können. Gleiches gilt auch für den Hobbyprogrammierer, der sich einmal an ein größeres Problem wagt.

**Die verbale formalisierte Beschreibung**

Im folgenden soll eine Form der verbalen formalisierten Beschreibung von Problemlösungen vorgestellt werden, die seit September 1983 im Rahmen eines »Fakultativen Kurses nach Rahmenplan« an der POS »Juri Gagarin« in Leipzig erfolgreich angewendet wird. Die Mitglieder dieses Kurses sind Schüler von 9. und 10. Klassen, die zu Beginn des Kurses über keinerlei Kenntnisse in der Informatik verfügten und anfangs sehr große Schwierigkeiten mit den streng logischen Arbeitsweisen der Informatik hatten.
Die Erläuterungen zur Methode erfolgen in diesem Abschnitt an Hand eines umfangreicheren Problems – wobei nicht alle Elemente verwendet werden –, das von den Schülern im September und Oktober des Jahres 1984 selbständig bearbeitet wurde und zwischenzeitlich als BASIC-Programm vollständig vorliegt. Es handelt sich dabei um folgende Aufgabe:
Es ist ein Programm zum Üben der vier Grundrechenarten zu erarbeiten. Die Übungsaufgaben sollen vom Rechner erstellt werden, wobei durch vorherige Eingabe die Rechenart, der Wertebereich und die Anzahl der zu generierenden Aufgaben vereinbart wurden. Die vom Schüler eingegebene Lösung der Aufgabe wird vom Computer überprüft und entsprechend kommentiert; erst nach zweimaliger falscher Lösung gibt der Rechner die richtige Antwort aus. Nach Abarbeitung der festgelegten Aufgabenanzahl gibt der Rechner das Ergebnis der Übung aus und fragt, ob weiter geübt werden soll. Wenn nicht, so soll sich das Programm verabschieden.
Die erste Überlegung zur Problemlösung ergibt der untenstehende Ausdruck.
In dieser ersten groben Lösung des Problems sind Namen und Schlüsselworte groß geschrieben und unterstrichen. Es treten dabei die Schlüsselworte BEGINN und ENDE auf, die den zu verarbeitenden Teil einklammern. In den restlichen Zeilen sind die Vereinbarungen und die eigentlichen Verarbeitungsschritte aufgeführt, wobei die Reihenfolge der einzelnen Zeilen des Verarbeitungsteils die Abarbeitungsfolge ergibt. Die einzelnen Zeilen sind hier als Teilalgorithmen zu sehen, die in den folgenden Lösungsschritten systematisch zu verfeinern sind. Auf jeder Stufe der Verfeinerung sollte jeweils eine Prüfung des erarbeiteten Teilalgorithmus erfolgen, da die Richtigkeit solcher Teile leichter zu ermitteln ist, als die des Gesamtalgorithmus.

```
RECHENTRAINER                              /* Programmname */
      Vereinbarungen
BEGINN                                /* Beginn Verarbeitungsteil */
      Ausgabe: Programmtitel
      Vorbereitung und Initialisierung
      Uebung
      Auswertung
      Endebehandlung
ENDE RECHENTRAINER
```

Die Verfeinerung des Teilalgorithmus »Ausgabe: Programmtitel« ist schnell abgeschlossen und ergibt:

```
AUSGABE: PROGRAMMTITEL
   BEGINN
      SCHREIB  "G R U N D R E C H E N A R T E N"
      SCHREIB  "=============================="
ENDE AUSGABE: PROGRAMMTITEL
```

Für den Teilalgorithmus »Vorbereitung« sind mehrere Verfeinerungsschritte notwendig, die hier gezeigt werden sollen.

```
VORBEREITUNG
    Ermittlung der gewuenschten Rechenart
    Ermittlung der gewuenschten Aufgabenanzahl
    Ermittlung des gewuenschten Wertebereichs
ENDE VORBEREITUNG
```

Im nächsten Schritt der Verfeinerung ermittelt man dann.

```
VORBEREITUNG
    Ausgabe: Menue fuer Auswahl der Rechenart
    Eingabe: gewuenschte Rechenart
    Ausgabe: Frage nach Aufgabenanzahl
    Eingabe: gewuenschte Aufgabenanzahl
    Ausgabe: Frage nach Wertebereich fuer die Operanden
    Eingabe: untere Grenze, obere Grenze
ENDE VORBEREITUNG
```

Letztendlich erhält man folgenden Teilalgorithmus:

```
VORBEREITUNG
  SCHREIB "WELCHE RECHENART WILLST DU RECHNEN?"
  SCHREIB "ADDITION=1              SUBTRAKTION=2"
  SCHREIB "MULTIPLIKATION=3        DIVISION=4"
  SCHREIB "VON JEDEM ETWAS=5"
  LIES  X                  /* Variable fuer die Rechenart */
  SCHREIB "WIE VIELE AUFGABEN WILLST DU RECHNEN?"
  LIES  A1                 /* Variable fuer Aufgabenzahl  */
  SCHREIB "KLEINSTE, GROESSTE ZAHL:"

  LIES   M1,M2             /* Variablen fuer  Wertebereich */
ENDE VORBEREITUNG
```

Die schrittweise Verfeinerung des Teilalgorithmus »Übung« ergibt folgende Bearbeitungsschritte.

```
UEBUNG
   BEGINN
      Startwert fuer Zufallszahlengenerator setzen
      Benoetigte Zaehler auf Null setzen
      FUER I=1 BIS Anzahl
      TUE
          Fehlerzaehler    =  0 setzen
          Ermittlung erster Operand
          Ermittlung zweiter Operand
          Verzweige entsprechend Operationsart
       ENDE
       Sprung zur Endebehandlung
ENDE UEBUNG
```

Die endgültige formale Problemlösung des Teilalgorithmus »Übung« hat dann folgendes Aussehen:

```
UEBUNG
   BEGINN
      Startwert fuer RND setzen
      R=0                    /* Zaehler richtige Loesungen */
      W=0                    /* Zaehler falsche Loesungen  */
      W1=0                   /* Zaehler Wiederholungen     */
      FUER I=1 BIS A1
      TUE  R1=0          /* Zeiger richtige Loesung    */
           F9=0              /* Fehlerzeiger                  */
           F1=INT(RND*D+M1)     /* Erster Operand          */
           F2=INT(RND*D+M1)     /* Zweiter Operand         */
           FALLS X1=5  DANN X=INT(RND*4+1)
           FALLS X=1   DANN  ADDITION
           FALLS X=2   DANN  SUBTRAKTION
           FALLS X=3   DANN  MULTIPLIKATION
           FALLS X=4   DANN  DIVISION
      ENDE
      Sprung zur Endebehandlung
ENDE UEBUNG
```

Bei Bejahung eines Tests mit $X = 1$ bis $X = 4$ wird das jeweilige Unterprogramm zur Generierung der entsprechenden Aufgabe, zur Ausgabe dieser Aufgabe am Bildschirm, Aufforderung der Lösungseingabe und Kontrolle der Eingabe, aufgerufen. Nach dem Test $X1 = 5$ wird die Berechnungsart für jede neue Aufgabe über den Zufallszahlengenerator ermittelt und dann über die folgenden Tests das entsprechende Unterprogramm aufgerufen.
Die Beispiele zeigen eindeutig, daß durch die verbale Kurzform die für Darstellungsmittel im Problemlösungsprozeß genannten Kriterien sehr gut erfüllt werden. Ein weiterer Vorteil dürfte in den einfachen Möglichkeiten der Änderung bzw. Ergänzung eines erstellten Algorithmus zu sehen sein. Wenn man beispielsweise in die verbale Kurzform des obigen Beispiels die Prüfung auf zulässige Eingabewerte einbauen will, so ist das sehr einfach zu realisieren. Man braucht nur auf der zugehörigen Verfeinerungsstufe die entsprechenden Anweisungen einzuschieben.

Weiterhin zeigen diese Beispiele, daß diese Darstellungsart unabhängig von einer bestimmten Programmiersprache ist, obwohl der versiertere Leser Ähnlichkeiten mit der höheren Programmiersprache PASCAL erkennen wird. Im folgenden soll gezeigt werden, mit welch geringem Aufwand die Umsetzung der formalen Problemlösung in eine höhere Programmiersprache – beispielsweise PASCAL oder BASIC – möglich ist.

**Programmerstellung**

Hat man sein zu lösendes Problem in der genannten Weise vollständig aufbereitet und diese Lösung mit einigen Werten auf logische und sachliche Richtigkeit und Vollständigkeit überprüft, so ist der nächste Schritt die Umsetzung der Problemlösung in eine dem Rechner verständliche Sprache. Zur Zeit bedeutet dies Programmieren in BASIC. Im Laufe der weiteren Entwicklung der Geräte – vor allem Vergrößerung der Hauptspeicher und die Nutzungsmöglichkeit von Diskettenlaufwerken – werden auch andere Spra-

chen zur Anwendung kommen, wie z. B. PASCAL oder LOGO.

*Programmerstellung in PASCAL*

Die schon genannte starke Ähnlichkeit der gezeigten verbalen Problemlösungsbeschreibung zu PASCAL ist der Grund, warum die Umsetzung in eine Programmiersprache zuerst am Beispiel dieser Sprache gezeigt werden soll.
Der Teilalgorithmus »Übung« könnte, in PASCAL programmiert, folgendes Aussehen haben:

```
PROCEDURE UEBUNG;
   BEGIN
      SWERT:=WERT;          (* Startwert fuer RND setzen *)
      R:=0;
      W:=0;
      W1:=0;
      FOR I:=1 TO A1 DO
      BEGIN
         R1:=0;
         F9:=0;
         F1:=TRUNC(RND(SWERT)*D+M1);
         F2:=TRUNC(RND(SWERT)*D+M1);
         IF X1=5 THEN X:=TRUNC(RND(SWERT)*X1+1)  ELSE X:=X1;
         IF X=1 THEN ADDITION;
         IF X=2 THEN SUBTRAKTION;
         IF X=3 THEN MULTIPLIKATION;
         IF X=4 THEN DIVISION
      END;
      ENDEBEHANDLUNG;
END; (*  UEBUNG  *)
```

Das vorstehende PASCAL-Programm ist eine möglichst direkte Umsetzung der verbalen Problemlösung. Man kann den Teilalgorithmus auch günstiger in PASCAL schreiben, aber darum geht es hier nicht. Der Zufallszahlengenerator RND ist in diesem Fall eine selbstprogrammierte Funktion, da es in PASCAL keine Standardfunktion für die Erzeugung von Zufallszahlen gibt. Der Funktion kann über SWERT ein beliebiger Startwert vorgegeben werden.

*Programmerstellung in BASIC*

Der Sprachumfang und die Ausdrucksmöglichkeiten von BASIC sind gegenüber PASCAL wesentlich eingeschränkt, so daß es nicht ganz so leicht ist, in einfacher und natürlicher Weise von einer verbalen Problemlösung zur Formulierung eines BASIC-Programms überzugehen. Das kann man auch aus der Gegenüberstellung der verbalen Problemlösung des Teilalgorithmus »Übung« und dem zugehörigen BASIC-Programm (s. S. 9 oben) entnehmen.

Der Programmtext entstammt einer Version des BASIC-Programms, das die Schüler im Dialog mit einer Rechenanlage ESER R 40 unter Verwendung des dort verfügbaren BASIC-Dialektes erstellt hatten. Dieser Dialekt hat noch geringere Möglichkeiten zur übersichtlichen Programmgestaltung als es bei den Kleincomputer-Dialekten der Fall ist.
Am Schluß des Artikels ist das vollständige BASIC-Programm des Schülerzirkels in der aktuellsten Version bei Verwendung eines Bürocomputers 1520, dessen BASIC mit dem Kleincomputer-BASIC vergleichbar ist, angegeben.

```
180 REM      ***  U E B U N G S T E I L  ***
190 REM
200 R=0
210 W=0
220 W1=0
222 FOR I=1 TO A1
224     R1=0
226     F9=0
230     F1=INT(RND*D+M1)
240     F2=INT(RND*D+M1)
245     IF X1<5 GOTO 260
250     X=INT(RND*X1+1)
260     IF X>1 GOTO 270
265     GOSUB 320
270     IF X<>2 GOTO 280
275     GOSUB 440
280     IF X<>3 GOTO 290
285     GOSUB 600
290     IF X<>4 GOTO 300
295     GOSUB 730
300 NEXT I
305 GOTO 870
308 REM     ***  ENDE UEBUNG  ***
```

Im Normalfall wird man der besseren Lesbarkeit wegen nur eine Anweisung pro Zeile schreiben. Hat man aber z. B. im Rahmen der Anfangswertzuweisung mehreren Variablen Werte zuzuweisen, so kann man auch mehrere Ergibtanweisungen – getrennt durch mindestens ein Leerzeichen – auf einer Zeile angeben.

## Die Elemente der verbalen Problemlösungsbeschreibung

In diesem Abschnitt wird eine vollständige Liste der Elemente der verbalen Problemlösungsbeschreibung angegeben. Diese Aufstellung gibt für jedes Element

- einen Namen
- die allgemeine Darstellungsform
- ein Beispiel (möglichst aus bisherigem Text)

an. In der allgemeinen Darstellungsform werden Elementeteile, die auftreten können, aber nicht unbedingt vorhanden sein müssen, in eckige Klammern gesetzt. Schlüsselworte, gültige Namen für Teilalgorithmen sowie Namen für Unterprogramme bzw. Funktionen werden in Großbuchstaben und allgemeingültige Bezeichnungen in Kleinbuchstaben geschrieben, wobei Schlüsselworte und Algorithmennamen zusätzlich noch unterstrichen werden.

### *Algorithmus bzw. Teilalgorithmus*

```
name
    [Vereinbarungen]
    BEGINN
        teile der algorithmenbeschreibung
ENDE name
```

*Beispiel:*

```
AUSGABE: PROGRAMMTITEL
    BEGINN
        SCHREIB ...
        SCHREIB ...
ENDE AUSGABE: PROGRAMMTITEL
```

Zwischen den Schlüsselworten
**BEGINN** und **ENDE**
steht der eigentliche Verarbeitungsteil des Algorithmus.

### *Vereinbarung*

Hier können die einzelnen Datentypen und Datenstrukturen in Typ und Aufbau definiert werden.

```
VEREINBARUNG

    art  name [,name]...  : typ
```

Bei »art« kann angegeben werden:

```
KONST    fuer Konstantenvereinbarung mit Anfangswertzu-
             weisung, ohne Typangabe

VAR      Variablenvereinbarung.
```

Für »typ« gibt es die Möglichkeiten:

```
GANZ          fuer Variable mit ganzzahligen Werten

REELL         fuer Variable mit reellen Werten

ZEICHENKETTE[1]     fuer Variable, die als Werte beliebige
                    Zeichen des Maschinenalphabets -- meist
                    Alphabetzeichen -- aufnehmen koennen.
                    Der Klammerwert ist die Angabe fuer
                    die maximale Zeichenzahl der Kette.
```

*Beispiele:*

```
VEREINBARUNG
     VAR  X,X1,A1,M1,M2,F1,F2   : GANZ
     VAR  A(10), B(2,3)         : REELL
     VAR  NAME                  : ZEICHENKETTE[10]
```

In der ersten Zeile werden eine ganze Reihe ganzzahliger Variablen vereinbart. In der zweiten Vereinbarungszeile wird die Variable A als Vektor mit 10 reellen Elementen und die Variable B als Matrix mit zwei Zeilen und 3 Spalten, deren Elemente reelle Werte annehmen können, vereinbart. Die letzte Vereinbarung definiert NAME als Zeichenkette, die maximal zehn Zeichen aufnehmen kann.

*Alternative*

Es gibt zwei Möglichkeiten der Alternative, die verkürzte und die vollständige.

```
FALLS bedingung DANN anweisung1
```

Die »anweisung 1« wird nur abgearbeitet, wenn die »bedingung« erfüllt ist. Im anderen Fall wird die nächste Anweisung in der Reihenfolge abgearbeitet.

Die vollständige Form der Alternative hat folgenden Aufbau

```
FALLS bedingung DANN anweisung1
                SONST anweisung2
```

Ist die »bedingung« erfüllt, so wird die »anweisung 1« abgearbeitet und anschließend die der Alternative folgende Anweisung. Bei Verneinung der »bedingung« wird erst die »anweisung 2« und dann die folgende Anweisung abgearbeitet.

*Beispiel:*

```
FALLS B=0 DANN   X=A
          SONST  X=A/B
```

Mit dieser Alternative wird abgesichert, daß die Division durch Null vermieden wird.

*Laufanweisung*

Die Laufanweisung ermöglicht die zyklisch wiederholte Abarbeitung einer Anweisungsfolge, wobei die Anzahl der Wiederholungen bei der ersten Abarbeitung der Laufanweisung schon festliegt.

```
FUER lv=w1 BIS w2   [STEP w3]
TUE
        anweisungen
ENDE
```

lv ist die Bezeichnung der Laufvariablen, die bei Abarbeitungsbeginn den Anfangswert w1 annimmt. Es werden die Anweisungen zwischen **TUE** und **ENDE** ausgeführt, wenn der Anfangswert w1 $<=$ w2, dem Endwert, ist. Mit Erreichen von **ENDE** wird die Laufvariable lv um die Schrittweite w3 ver-

ändert. Standardmäßig ist die Schrittweite w3 = 1 und braucht dann nicht angegeben zu werden. Soll die Schrittweite einen anderen Wert als "+1" haben, so muß der neue Wert zusammen mit dem Schlüsselwort **STEP** angegeben werden. Es ist auch eine negative Schrittweite möglich. Im Teilalgorithmus »Übung« ist eine Laufanweisung enthalten. Die Laufvariable heißt dort I und erhält den Anfangswert 1. Die Anweisungen zwischen **TUE** und **ENDE** werden so oft abgearbeitet, bis die Laufvariable I den Endwert A1 überschritten hat. Die Schrittweite ist die Standardschrittweite 1.

*Wiederholung bzw. Schleife*

Die Schleife ermöglicht die bedingte Wiederholung einer Anweisungsfolge. Es gibt zwei Möglichkeiten für die Darstellung der Wiederholung:

– die nichtabweisende Schleife

```
WIEDERHOLE
          anweisungen
BIS bedingung
```

Diese Schleifenart wird mindestens einmal abgearbeitet, da die »bedingung« erst am Schleifenende auf Erfüllung getestet wird. Ist die »bedingung« nicht erfüllt, so wird die Anweisungsfolge erneut abgearbeitet.

– die abweisende Schleife

```
SOLANGE bedingung
TUE
        anweisungen
ENDE
```

Ist »Bedingung« bei Eintritt in die Schleife nicht erfüllt, so wird die ganze Schleife nicht ausgeführt, sondern die Anweisung nach **ENDE**. Im anderen Fall wird die Schleife so oft abgearbeitet, bis die »bedingung« nicht mehr erfüllt ist.

Im Gegensatz zur Laufanweisung steht die Anzahl der Durchläufe bei Eintritt in die Schleife noch nicht fest. Der Programmierer muß bei der Erarbeitung des Algorithmus dafür Sorge tragen, daß die »bedingung« innerhalb der Schleife so verändert wird, daß in endlicher Zeit die »bedingung« nicht mehr erfüllt ist und die Programmabarbeitung nach der Schleife fortgesetzt werden kann.

*Beispiele:*

```
ZAEHLER=0     SUMME=0
WIEDERHOLE
        SUMME = SUMME + ZAEHLER * 2
        ZAEHLER = ZAEHLER + 2
BIS ZAEHLER=100
```

```
ZAEHLER=0     SUMME=0
SOLANGE ZAEHLER<101
TUE
        SUMME = SUMME + ZAEHLER *2
        ZAEHLER = ZAEHLER + 2
ENDE
```

Beide Schleifen haben den gleichen Effekt, sie werden jeweils 50mal durchlaufen, und SUMME hat nach Abarbeitung den Wert 4900.

*Eingabe*

```
LIES [datei] name [, name]...
```

Im Teilalgorithmus »Vorbereitung« wurde die Eingabeanweisung in der Form

```
LIES X1        bzw.    LIES M1,M2
```

verwendet. Diese Anweisungen bewirken, daß vom Programm der Wert für X1, der über Tastatur eingegeben wurde, gelesen und abgespeichert wird und im zweiten Fall die eingegebenen Werte für M1 und M2. Die Eingabe über die Tastatur wird als die Standardform verwendet und erfordert deshalb nicht die Angabe 'datei'. Diese Angabe ist notwendig, wenn zur Lösung

des Problems Werte aus externen Dateien, die auf Kassette bzw. Diskette abgespeichert sind, eingelesen werden müssen.

*Ausgabe*

Die allgemeine Form der Ausgabeanweisung lautet

```
SCHREIB [datei] ausgabeliste
```

wobei für 'datei' gilt, daß der Bildschirm die Standardausgabe ist. Sollen Werte auf Drucker oder externe Speichermedien ausgegeben werden, so muß hier der entsprechende Dateiname angegeben werden.
'ausgabeliste' kann – wie in den obigen Beispielen angewendet – ein einzelner in Apostrophe eingeschlossener Text, ein einzelner Variablenname oder mehrere dieser genannten Elemente, die durch Kommas getrennt aufgelistet sind, sein. In Apostrophen stehende Zeichen werden so ausgegeben, wie sie in der 'ausgabeliste' stehen. Statt der Variablennamen in der 'ausgabeliste' werden die aktuellen Werte dieser Variablen am Bildschirm angezeigt bzw. auf die entsprechende Datei ausgegeben.

*Kommentar*

Kommentare sind Bemerkungen, die die Lesbarkeit der jeweiligen Darstellungsform erhöhen sollen. Wie den obigen Beispielen zu entnehmen ist, hat der Kommentar die Form

```
/*   beliebiger text    */
```

und kann sowohl nach einer Anweisung oder allein auf einer Zeile stehen. 'beliebiger text' heißt, daß alle für die Maschine definierten Zeichen an dieser Stelle erlaubt sind.

*Ergibtanweisungen und Ausdrücke*

Für Ergibtanweisungen und Ausdrücke wurde keine neue Schreibweise festgelegt. Es sollte im wesentlichen die in der Mathematik bzw. Logik übliche Darstellungsform verwendet werden. Allerdings ist es sinnvoll, in mathematischen Ausdrücken

- $*$ als Multiplikationszeichen
- $/$ als Divisionszeichen
- $**$ als Potenzzeichen

zu verwenden. Im Gegensatz zur üblichen mathematischen Schreibweise sollte man das Multiplikationszeichen immer schreiben, um die Eindeutigkeit zu wahren. So sollte man den mathematisch exakten Ausdruck

$$2\,a\,(b + 3\,c)$$

in der verbalen Kurzform lieber in der Form

$$2 * A * (B + 3 * C)$$

schreiben, wie es auch bei Ausdrücken in den meisten höheren Programmiersprachen verlangt wird.

**BASIC-Programm**

Das BASIC-Programm (s. S. 13) enthält noch einige Erweiterungen, die in den einführenden Beispielen zur Darstellung der formalen Problemlösung nicht gezeigt wurden. Diese Erweiterungen wurden von den Schülern während der Programmtestung und -erprobung nach und nach eingefügt. Der Programmtext wurde durch entsprechende Kommentare noch lesbarer und verständlicher gestaltet.

Autor:

*Dr.-Ing. Alfred H. Schilling*

Gruppenleiter Systemprogrammierung
im ORZ der Karl-Marx-Universität Leipzig

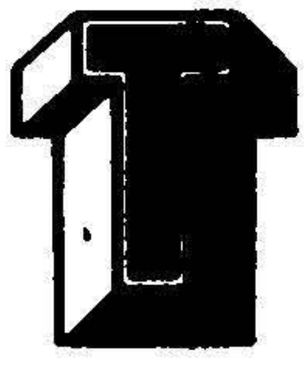
```
10 REM  ************************************************************
12 REM  **          R E C H E N T R A I N I N G                   **
14 REM  **        ==============================                  **
16 REM  **           FKR 'INFORMATIK' DER POS                     **
18 REM  **           "JURI GAGARIN"  LEIPZIG                      **
20 REM  **                JANUAR 1985                             **
22 REM  ************************************************************
25 PRINT "  G R U N D R E C H E N A R T E N"
30 PRINT " ================================="
35 PRINT
40 REM          *****     VORBEREITUNGEN     *****
42 REM          ********************************
45 U=0          :  REM  Zaehler fuer Gesamtaufgabenanzahl
46 L=0          :  REM  Schalter fuer Druck Bewertung
47 R=0          :  REM  Zaehler richtiger Loesungen
50 PRINT "Welche Rechenart willst Du ueben?"
60 PRINT "Addition       = 1        Subtraktion = 2"
70 PRINT "Multiplikation = 3        Division    = 4"
80 PRINT "Von jedem etwas= 5"
90 INPUT X      :  REM  Rechenarteneingabe
95 PRINT
100 IF (X>5) OR (X<1) THEN GOSUB 1410  :  GOTO 50
110 PRINT " Wie viele Aufgaben willst Du rechnen?"
120 INPUT A1    :  REM  Eingabe Aufgabenanzahl
125 PRINT
130 IF (A1<1) OR (A1>INT(A1)) THEN GOSUB 1410  :  GOTO 110
135 U=U+A1      :  REM  Stellen Aufgabenzaehler
140 PRINT "Kleinste, groesste Zahl:"
150 INPUT M1,M2  :  REM  Eingabe Wertebereich
155 PRINT
160 IF (M1<INT(M1)) OR (M2<INT(M2)) THEN GOSUB 1410 : GOTO 140
170 IF M2<M1 THEN Z=M1 : M1=M2 : M2=Z : REM Grenzen vertauschen
180 IF (X=2) OR (X=5) GOTO 200
185 IF X=4 GOTO 240
190 GOTO 300
200 PRINT "Negative Differenz zulaessig (ja=1/nein=2)?"
210 INPUT Y
220 IF (Y<1) OR (Y>2) THEN GOSUB 1410  :  GOTO 200
230 PRINT
235 IF X<4 GOTO 300
240 PRINT "Auf wieviel Nachkommastellen genau (0 bis 6)?"
250 INPUT K
260 IF (K<0) OR (K>6) THEN GOSUB 1410  :  GOTO 240
270 PRINT
275 REM     *****     ENDE VORBEREITUNGEN     *****
280 REM
290 REM     *****     U E B U N G S T E I L     *****
300 REM     ************************************************
310 RANDOMIZE (M2)   :  REM Startwert Zufallszahlengenerator
320 D=M2-M1+1  :  X1=X
330 FOR I=1 TO A1    :  REM  Laufanweisung fuer Uebung
350 .   F9=0              :  REM  Fehlerzaehler
360     F1=INT(RND*D+M1)  :  REM  Erster Operand
370     F2=INT(RND*D+M1)  :  REM  Zweiter Operand
380     IF (F2=0) AND (X>3) GOTO 370
390     IF X1=5 THEN X=INT(RND*4+1) : REM Rechenart waehlen
400     IF X=1 GOTO 440   :  REM  Aufruf ADDITION
410     IF X=2 GOTO 500   :  REM  Aufruf SUBTRAKTION
420     IF X=3 GOTO 570   :  REM  Aufruf MULTIPLIKATION
430     GOTO 630          :  REM  Aufruf DIVISION
440     REM   ***   A D D I T I O N   ***
445     REM   ************************
450     PRINT F1;" + ";F2;" = ";
460     INPUT S           :  REM  Eingabe Schuelerloesung
```

```
470     P=F1+F2              :  REM  Ermittl. Loesung
480     GOTO 800             :  REM  Sprung zum Vergleich
490     REM   ***   ENDE ADDITION   ***
500     REM   ***   S U B T R A K T I O N   ***
502     REM   ***************************************
505     REM  Eventuelles Vertauschen der Operanden
510     IF (Y=2) AND (F1-F<20) THEN Z=F1 : F1=F2 : F2=Z
520     PRINT F1;" - ";F2;" = ";
530     INPUT S
540     P=F1-F2
550     GOTO 800
560     REM   ***   ENDE SUBTRAKTION .  ***
570     REM   ***   M U L T I P L I K A T I O N   ***
575     REM   ***************************************
580     PRINT F1;" * ";F2;" = ";
590     INPUT S
600     P=F1*F2
610     GOTO 800
620     REM   ***   ENDE MULTIPLIKATION   ***
630     REM   ***   D I V I S I O N   ***
635     REM   ***************************************
640     PRINT F1;" : ";F2;" = ";
650     INPUT S
660     P=F1/F2
665     REM   ***   Ermittlung der Dezimalstellen   ***
667     REM   ***   mit waehlbarer Stellenzahl   ***
670     Q1=INT(P*10^(K+1))
680     Q2=INT(P*10^K)*10
690     Q3=Q1-Q2
700     IF Q3<5 THEN Q4=0     ELSE Q4=1
710     Q5=INT(P*10^K+Q4)/10^K
720     Q6=INT(S*10^(K+1))
730     Q7=INT(S*10^K)*10
740     Q8=Q6-Q7
750     IF Q8<5 THEN Q9=0     ELSE Q9=1
760     C=INT(S*10^K+Q9)/10^K
770     IF (C>Q5+.5*10^(-K)) OR (C<Q5-.5*10^(-K)) GOTO 800
780     S=P
790     REM   ***   ENDE DIVISION UND RUNDEN   ***
800     REM   ***   VERGLEICH DER LOESUNGEN   ***
805     REM   ***************************************
810     IF (S=P) AND (F9=0) THEN PRINT "Richtig!" : R=R+1 : GOTO 910
820     IF (S=P) AND (F9>0) GOTO 890
830     PRINT "Leider falsch!"
840     PRINT "Willst Du es noch einmal versuchen (ja=1/nein=2)?"
850     INPUT G
860     IF G=1 THEN F9=F9+1  :  GOTO 400  : REM Gleiche Aufgabe
870     IF G=2 THEN PRINT "Das richtige Ergebnis lautet:";P :GOTO 910
880     GOSUB 1410   :  GOTO 840  :  REM  Bei falscher Eingabe
890     PRINT "Jetzt war es richtig!"
900     R=R+1/(F9+1)

910 NEXT I       :  REM  Ende Laufanweisung fuer Uebung
920 REM      *****      ENDE UEBUNGSTEIL      *****
925 REM      ***************************************
930 REM      *****      A U S W E R T U N G      *****
935 REM      ***************************************
940 PRINT "Auswertung gewuenscht (ja=1/nein=2)?"
950 INPUT H
960 IF (H<1) OR (H>2) THEN GOSUB 1410  :  GOTO 940
970 IF H=2 GOTO 50     :  REM Sprung zum erneuten Durchlauf
975 IF L=1 GOTO 1060   :  REM Druck Bewertungssystem uebersp.
980 PRINT "Bewertungssystem:"
990 PRINT "Beim ersten Mal richtig geloeste Aufgabe : je  1  Punkt"
```

```
1000 PRINT "Beim zweiten Mal richtig geloeste Aufgabe: je 1/2 Punkt"
1010 PRINT "Beim dritten Mal richtig geloeste Aufgabe: je 1/3 Punkt"
1020 PRINT "Beim vierten Mal richtig geloeste Aufgabe: je 1/4 Punkt"
1030 PRINT "usw."
1040 PRINT
1060 PRINT "Erreichte Punktzahl   :";R
1070 PRINT "Erreichbare Punktzahl :";U
1080 PRINT
1090 V=100*R/U
1100 PRINT "Loesungsquote:  ";V;" %"
1110 PRINT
1120 IF V>=96 GOTO 1170
1130 IF V>=80 GOTO 1190
1140 IF V>=60 GOTO 1210
1150 IF V>=36 GOTO 1230
1160 GOTO 1250
1170 PRINT "Dein Lehrer wuerde Dir jetzt eine 1 geben."
1180 GOTO 1300
1190 PRINT "Dein Lehrer wuerde Dir jetzt eine 2 geben."
1200 GOTO 1300
1210 PRINT "Dein Lehrer wuerde Dir jetzt eine 3 geben."
1220 GOTO 1300
1230 PRINT "Dein Lehrer wuerde Dir jetzt eine 4 geben."
1240 GOTO 1300
1250 PRINT "Dein Lehrer wuerde Dir jetzt eine 5 geben,"
1260 PRINT "wuetend die Zensur einschreiben und sagen:"
1270 PRINT "'Das hast Du Dir selbst zuzuschreiben!'"
1275 PRINT "'Sieh zu, wie Du das wieder ausbuegelst!'"
1280 PRINT "Ich, der Computer, meine, dass Du froh sein kannst,"
1285 PRINT "dass das nur eine Uebung war!"
1290 PRINT "Uebe nur fleissig weiter, jetzt geht es noch!"
1295 PRINT "Trotzdem:  S c h a e m   D i c h !!!!!"
1300 PRINT
1305 REM   *****      E N D E B E H A N D L U N G      *****
1307 REM   ***************************************************
1310 PRINT "Neubeginn (ja=1/nein=2)?"
1320 INPUT N
1330 IF (N<1) OR (N>2) THEN GOSUB 1410  :  GOTO 1310
1340 IF N=1 GOTO 1360
1350 IF N=2 GOT 1460
1360 PRINT "Punkte weiter zaehlen (ja=1/nein=2)?"
1370 INPUT L
1380 IF (L<1) OR (L>2) THEN GOSUB 1410  :  GOTO 1360
1385 PRINT
1390 IF L=1 GOTO 50      ELSE GOTO 40
1400 REM     *****      ENDE AUSWERTUNG      *****
1405 REM   ***  Unterprogramm zur Behandlung von Fehleingaben   ***
1407 REM   ***************************************************
1410 PRINT
1420 PRINT "Falsche Eingabe!"
1430 PRINT
1440 RETURN
1450 REM   ***   Ende Unterprogramm   ***
1460 REM   *****      Endebehandlung      *****
1465 REM   ***************************************
1470 PRINT
1480 PRINT "Auf baldiges Wiedersehen!"
1490 PRINT
1500 END
```

# Wieso rechnet (m)ein Taschenrechner $\sqrt{8} \cdot \sqrt{8} = 8$ ?



Die Frage der Überschrift scheint für Kundige der Schulmathematik keine zu sein. Schließlich ist die Quadratwurzel so definiert, daß sie, mit sich selbst multipliziert, den Radikanden ergibt. Allerdings müssen hierzu im allgemeinen alle unendlich vielen Stellen der Wurzel bemüht werden, die sie als irrationale Zahl besitzt.
Genau hier liegt das Problem des Taschenrechners und der Kleincomputer; sie müssen alles mit wesentlich weniger – häufig mit nur fünf bis acht – Stellen machen. Jede Stellenabschneidung muß aber zur Wurzelverkleinerung und daraus resultierend auch zur Produktverringerung im genannten Problem führen; es muß also gelten:

$(\sqrt{8}_{\text{stellenbegrenzt}})^2 < 8$, z. B. in der Form

$(\sqrt{8}_{\text{8stellig}})^2 = (2{,}8284271)^2_{\text{8stellig}}$
$= 7{,}9999998.$

Betrachten wir im weiteren die Probleme stellvertretend an Hand der Taschenrechner. Einfache oder ältere Modelle, bei denen die Stellenzahl der Anzeige mit der Rechnungs-Stellenzahl übereinstimmt, liefern derartige Resultate auch – so z. B. die Uhrenrechner der DDR-Produktion.
Wie erzeugen moderne Taschenrechner dennoch die glatte Anzeige – 8 – so lautet nun die inzwischen als berechtigt anerkannte Fragestellung.

Leistungsfähigere Modelle können über die Stellen der Anzeige hinaus mit mindestens einer weiteren, sogenannten verdeckten Rechenstelle arbeiten, was z. B. die folgende Operation in Form der verbleibenden Drei(en) sichtbar macht:

| 1 | 0 | ÷ | 3 | = | ... |
|---|---|---|---|---|---|

| – | Anzeigewert | = | ... |
|---|---|---|---|

Mit Hilfe dieser verdeckten Stelle(n) kann auf mindestens drei im weiteren beschriebenen, modellspezifischen Wegen die anvisierte Anzeige- bzw. Ergebnis-Nachbesserung erreicht werden.

1. Zunächst sollen Taschenrechner-Modelle betrachtet werden wie MR 610 und MR 609 (SR 1), die ihre einzige verdeckte Stelle zur **Anzeigerundung nach einfachsten mathematischen Regeln** nutzen; (0 · · · 4 ⇝ keine Änderung der vorhergehenden Anzeigestelle, 5 bis 9 ⇝ Erhöhung der vorhergehenden Anzeigestelle um 1).
Bestätigung hierfür liefern die Beispielrechnungen

| 1 | 0 | ÷ | 3 | × | 2 | = | 6,6666667 |
|---|---|---|---|---|---|---|---|

| 1 | + | 5 | EEX | 8 | +/– | = |
|---|---|---|---|---|---|---|

1,0000001,

angegeben für Rechner mit 8 Anzeigestellen (EEX = Exponenteneingabe; +/– = Vorzeichenwechsel).

Von einer Anzeigerundung wird gesprochen, weil eventuelle Rechnungsfortsetzungen von dem ungerundeten Wert ausgehen. Hiervon kann man sich durch Subtraktion jeweils des Anzeigewertes in den letzten beiden Beispielen überzeugen, woraufhin die fehlenden 4 bzw. 5 Einheiten der verdeckten Stelle als negative Differenz ausgegeben werden.
Während eine eventuelle Anzeigerundung der Wurzel (vom zufälligen Wert abhängig) deren Quadrat also nicht beeinflussen kann, führt die mit Sicherheit auftretende Quadratrundung zum Auftauchen des originären Radikanden in der Anzeige:

$\sqrt{8}_{(8+1)\text{-stellig}} = 2{,}8284271$ [2 verdeckt]

$\left(\sqrt{8}_{(8+1)\text{-stellig}}\right)^2 = 8$ laut Anzeige.

Die Fortsetzung der Quadrierung wieder durch Subtraktion des Anzeigewertes liefert die für diese Problemlösung charakteristische negative Differenz (hier $-3 \cdot 10^{-8}$). Dahinter verbirgt sich der Hinweis, daß der Rechner eigentlich

$\left(\sqrt{8}_{(8+1)\text{-stellig}}\right)^2{}_{(8+1)\text{-stellig}} =$

7,9999999 [7 verdeckt]

herausbekommen hat und nur diesen Wert für eventuelle Weiterrechnungen bereithält.

2. Jetzt sollen Modelle betrachtet werden, die mit ihrer einzigen verdeckten Stelle **keine Rundung** vornehmen, **aber** das Themenproblem durch eine $\sqrt{}$**-Vergrößerung** lösen. Die fehlende Rundung zeigt z.B. die ungestörte Periodizität in Sechsen des Rechnungsergebnisses

| 2 | 0 | ÷ | 3 | = |
|---|---|---|---|---|

6,6666666 [6 verdeckt].

Verschafft man sich auf derartigen Rechnern (z.B. COMPEX 5500) Aufschluß über $\sqrt{8}$, einschließlich der verdeckten Stelle, erhält man

$\sqrt{8}_{(8+1)\text{-stellig}} = 2{,}8284271$ [3 verdeckt].

Es liegt eine gewollte, aber durch keine Rundungsregel begründbare Abweichung vom genauen Wurzelwert[1]) vor. Wieder liefert die Quadrierung dadurch $\left(\sqrt{8}_{(8+1)\text{-stellig}}\right)^2 = 8$ laut Anzeige.
Die Subtraktion der Anzeige 8 führt jetzt zu einer charakteristischen positiven Differenz (hier $2 \cdot 10^{-8}$).
Erneut ist also der korrekte Radikand nur zu sehen bzw. für unmittelbare Ausgaben bereitgestellt, es wird aber mit dem abweichenden Wert

$\left(\sqrt{8}_{(8+1)\text{-stellig}}\right)^2{}_{(8+1)\text{-stellig}} =$

8,0000000 [2 verdeckt]

gegebenenfalls weitergerechnet.
Vergleiche zwischen Berechnungszeiten von Quadratwurzeln und Wurzeln mit anderen Exponenten $\left(\sqrt[x]{y}\right)$ lassen darauf schließen, daß sich einmal schnell konvergierende Iterationsverfahren einsetzen lassen, während zum anderen auf zeitaufwendige Logarithmenrechnungen zurückgegriffen werden muß.
Das zur Berechnung von $W = \sqrt{R}$ einsetzbare NEWTONsche Iterationsverfahren mit $W_{n+1} = \dfrac{\frac{R}{W_n} + W_n}{2}$ zeigt Bild 1 als Programmablaufplan. Es folgt die Ergebniszusammenstellung für $R = 8$; der Radikand $R$ selbst dient zugleich als erster Schätzwert $W_0$ für die Wurzel:

$W = \sqrt{R} = ?\quad R = 8$
$W_0 = 8$
$W_1 = 4{,}5$
$W_2 = 3{,}138\ 8888$
$W_3 = 2{,}843\ 7807$

---

[1]) $\sqrt{8}_{12\text{stellig}} = 2{,}82842712474\ldots$

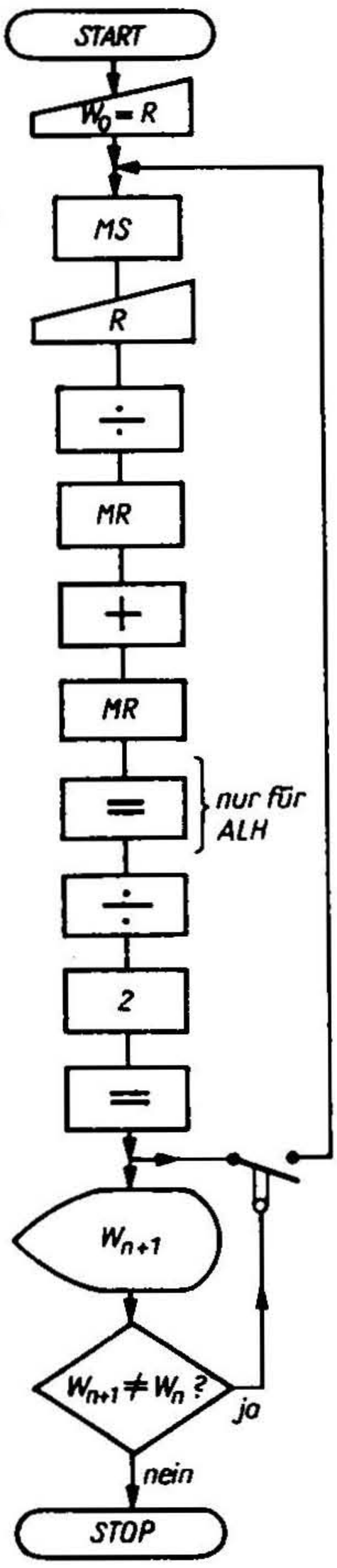



Bild 1
Iterative Quadratwurzelberechnung (8stelliger Rechner ohne Rundung)

MS Einspeicherkommando
MR Speicherrückrufkommando
ALH algebraische Logik mit Hierarchie

$W_4 = 2{,}828\ 4685$
$W_5 = 2{,}828\ 4271$
$W_6 = 2{,}828\ 4271$
$W = \sqrt{8} = W_5 = W_6$

Beim »Durchspielen« des Programms von Hand mit dem Taschenrechner – anstelle einer Nutzung in Form der festverdrahteten Funktion $\sqrt{x}$ – wird häufig die Verzweigungsfrage extern entschieden werden müssen (daher die etwas eigenwillige Darstellung der Programm-Weiterführung).

Die hier interessierenden $\sqrt{}$-Aufschläge – im Beispiel oben von 1 in der verdeckten Stelle – lassen sich durch Programmänderungen realisieren wie Aufnahme einer zusätzlichen Addition für den letztgültigen Wert oder Modifizierung des Abbruchkriteriums (Wertübereinstimmung nur in Anzeigestellen).

3. Abschließend sollen Modelle betrachtet werden, die mit drei verdeckten Stellen eine **endgültige Ergebnisglättung** vornehmen. Die verdeckten Stellen mit ihren Werten 000···999 können nicht nur wie üblich auf Überschreitung einer vorgegebenen Grenze für eventuelle Anzeigerundungen abgefragt werden, sondern bei einigen Taschenrechner-Modellen werden die Werte 001 bis 009 z.B. endgültig gelöscht zugunsten einer irreversiblen Ergebnisverkleinerung, während die Werte 990 bis 999 z.B. zu einer irreversiblen Ergebnisvergrößerung (+ 1000) genutzt werden (Vertreter z.B. SANYO CZ 1201). Bei der $\sqrt{}$-Quadrierung steht damit der korrekte Radikand $R$ auch zur Weiterverarbeitung bereit; Subtraktion einer 8 vom Anzeigewert 8 ergibt hier exakt Null.

$(\sqrt{8}_{(8+3)\text{-stellig}})^2_{\text{geglättet}} \equiv 8$

Das 11stellige eigentliche $\sqrt{}$-Produkt vor seiner Glättung läßt sich mit einem Taschenrechner unter Zuhilfenahme der binomischen Formel $(a+b)^2 = a^2 + 2ab + b^2$ ermitteln; (abschließende Addition von Hand!).

$\sqrt{8}_{(8+3)\text{-stellig}} = 2{,}8284271\ [247\ \text{verdeckt}]$
$= a + b = 2{,}828 + 4{,}271247 \cdot 10^{-4}$

| | | |
|---|---|---|
| $a^2 =$ | 7,9975840 | 000 |
| $2ab =$ | 0,0024158 | 173 |
| $b^2 =$ | 0,0000001 | 824 |
| $\Sigma =$ | 7,9999999 | 997 $= (\sqrt{8}_{(8+3)\text{-stellig}})^2$ |

Die verdeckten $997 \geqq 990$ lösen die Addition von 1000 aus, lassen das glatte Ergebnis 8 entstehen.

Diese Ergebnisglättung, die das betrachtete Problem ideal löst, ist dennoch nicht unumstritten. Auch andere Zahlen bzw. Rechnungsergebnisse, die zufällig diese verdeckten Stellenwerte aufweisen bzw. liefern, werden in gleicher Weise – aber unbegründet – »geglättet«. Dieser Gedankengang soll mit nachstehender Beispielrechnung verdeutlicht werden:

[1] [÷] *n* · [9] [EEX] [1] [0] [+/-] [=] ...

(*n* positiv ganze Zahl)

Bei Berechnung des Ausdrucks über *n*-malige Addition von $9 \cdot 10^{-10}$ wird in jeder Zwischensumme der Addend (2. Summand) »weggeglättet« und als Ergebnis erscheint der unveränderte Augend 1 (1. Summand).

Autor:

*Dr. rer. nat. Wolfgang Michel*

Dozent an der Ingenieurhochschule Wismar

## Beherrschen Sie BASIC?

Wenn ja, dann wird es Ihnen keine Schwierigkeiten bereiten, die folgenden kleinen *Denkaufgaben* zu lösen. Wenn nein, dann finden Sie auf Seite 58 dieses Heftes Hinweise, die Ihnen weiterhelfen werden.

**Aufgabe 1:** Was druckt der Rechner aus und wieviel Zyklen werden durchlaufen, wenn das folgende BASIC-Programm eingegeben und gestartet wird:

```
10  CLS
20  FOR I=1 TO 100
30  I=I-1
40  PRINT I;
50  NEXT I
60  END
```

**Aufgabe 2:** Wozu könnte das folgende BASIC-Programm wohl verwendet werden?

```
10  CLS
20  FOR K=1 TO 5
30  L=INT(35*RND(K)+1)
40  PRINT L;
50  NEXT K
60  PRINT
70  END
```

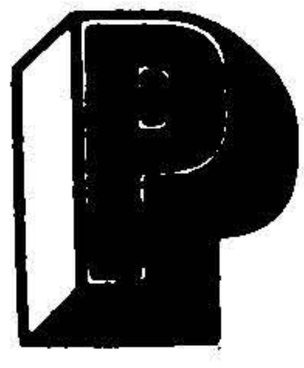
**Aufgabe 3:** In dem folgenden Programm treten unterschiedliche Variablen auf, die alle mit *dem gleichen Namen* I bezeichnet werden:

| | |
|---|---|
| arithmetische Größen | I, |
| Zeichenketten | I$, |
| ein Feld von arithmetischen Größen | I(10) und |
| ein Feld von Zeichenketten | I$(10). |

Ist der Rechner in der Lage, diese unterschiedlichen Datentypen mit gleicher Bezeichnung auseinanderzuhalten?

```
10  CLS
20  I$="HANS"
30  FOR I=1 TO 10
40  I(I)=I+5
50  I$(I)=CHR$(64+I)
60  PRINT TAB(5); I; TAB(10);
    I$; TAB(16); I(I);
    TAB(21); I$(I)
70  NEXT I
80  END
```

# Kalenderfunktionen



Mit dem Kleincomputer lassen sich auf einfache Weise Programme erstellen, die Funktionen eines elektronischen Notizbuches, eines Terminkalenders u.ä. realisieren. In solchen Anwendungen spielt das Datum der betreffenden Eingaben eine wesentliche Rolle. Neben dem Datum selbst ist vor allem für Terminkalender der betreffende Wochentag von Interesse.
Das im folgenden vorgestellte BASIC-Programm »KALENDERFUNKTIONEN« ermöglicht die Beantwortung verschiedener mit dem Datum zusammenhängender Fragen. Die folgenden Fragestellungen wurden berücksichtigt:

- Auf welchen Wochentag fällt ein bestimmtes Datum?
- Wieviel Tage liegen zwischen zwei Daten?
- Welches Datum erhält man, ausgehend von einem Ausgangsdatum und einer bestimmten Anzahl von Tagen zwischen dem zu bestimmenden und dem Ausgangsdatum?

Der Hintergrund für die zugrunde liegenden Berechnungen liegt in der Periodizität des gültigen Kalenders. Für den interessierten Leser sei an dieser Stelle eine Möglichkeit zum Nachschlagen genannt. In [1] sind neben Informationen zu Grundlagen und Gültigkeitsdauer der verschiedenen Kalender Tabellen enthalten, die, wenn auch recht umständlich, eine Beantwortung der oben angeführten Fragestellungen zulassen.
Bevor das Programm als solches vorgestellt wird, soll die Handhabung des Programms erläutert werden. Zur Auswahl einer der drei möglichen Kalenderfunktionen wird auf dem Bildschirm ein sogenanntes Menü, welches die Auswahlmöglichkeiten und die erforderlichen Eingaben darstellt, aufgebaut. Nach erfolgter Auswahl wird der für die betreffende Funktion erforderliche Dialog zur Eingabe der notwendigen Daten geführt. Nach der Ausgabe des Ergebnisses erscheint die Frage nach Wiederholung der Programmabarbeitung. Je nach Beantwortung dieser Fragestel-

```
**********  KALENDER FUNKTIONEN  *****

 <1>    BESTIMMUNG DES WOCHENTAGES
 <2>    BESTIMMUNG DER TAGESDIFFERENZ
        ZWISCHEN ZWEI DATEN
 <3>    BESTIMMUNG EINES DATUMS AUS
        AUS DATUM UND TAGESDIFFERENZ

TASTEN SIE DIE NR. DER GEWUENSCHTEN
ROUTINE [1 oder 2 oder 3] EIN !

?1

DATUM :
TAG  ?4
MONAT?3
JAHR ?85

4.3.1985          MONTAG

WIEDERHOLEN? [J/N]?
```

Bild 1. Bildschirminhalt bei der Bestimmung des Wochentages

```
********** KALENDER FUNKTIONEN ******

  1>    BESTIMMUNG DES WOCHENTAGES
  2>    BESTIMMUNG DER TAGESDIFFERENZ
        ZWISCHEN ZWEI DATEN
  3     BESTIMMUNG EINES DATUMS AUS
        AUS DATUM UND TAGESDIFFERENZ

TASTEN SIE DIE No. DER GEWUENSCHTEN
ROUTINE [1 oder 2 oder 3] EIN !

JEDE EINGABE MIT [ENTER] ABSCHLIESSEN !

?2

1. DATUM :
TAG  ?23
MONAT?3
JAHR ?52

2. DATUM :
TAG  ?4
MONAT?3
JAHR ?85

1. DATUM :  23.3.1952
2. DATUM :  4 .3.1985

DIE DIFFERENZ ZWISCHEN DEN DATEN BETRAEGT
12034    TAGE

WIEDERHOLEN? [J/N]?
```

Bild 2. Bildschirminhalt bei der Bestimmung der Differenz zwischen zwei Tagen

```
********** KALENDER FUNKTIONEN ******

 <1>    BESTIMMUNG DES WOCHENTAGES
 <2>    BESTIMMUNG DER TAGESDIFFERENZ
        ZWISCHEN ZWEI DATEN
 <3>    BESTIMMUNG EINES DATUMS AUS
        AUS DATUM UND TAGESDIFFERENZ

TASTEN SIE DIE No. DER GEWUENSCHTEN
ROUTINE [1 oder 2 oder 3] EIN !

JEDE EINGABE MIT [ENTER] ABSCHLIESSEN

?3

DATUM :
TAG  ?23
MONAT?3
JAHR ?52

AUSGANGSDATUM :  23.3.1952

TAGESDIFFERENZ: 12034

ZIELDATUM :      4.3.1985

WIEDERHOLEN? [J/N]?
```

Bild 3. Bildschirminhalt bei der Bestimmung des Datums aus Datum und Tagesdifferenz

lung erfolgen Neustart oder Stop des Programms.
Die Bilder 1 bis 3 zeigen den Bildschirminhalt für die drei wählbaren Funktionen. Das Menü ist natürlich für alle Funktionen gleich. In Bild 4 ist die Liste des Programms ausgegeben. Damit die Programmfunktionen auch in anderen Anwendungen genutzt werden können, wurde eine Reihe von weiterverwendbaren Unterprogrammen geschaffen. Die das Unterprogramm jeweils kommentierende REM-Zeile wurde so angelegt, daß der Unterprogrammeinsprung stets nach dieser Zeile erfolgt. Für die vorliegende Anwendung ist dies sicher wenig von Belang, doch wird die nicht erforderliche Übersetzung umgangen. Beginnt ein Unterprogramm z.B. mit der Zeile 300, so steht die kommentierende REM-Zeile bei 299. Im folgenden sind die verwendeten Unterprogramme mit ihren entsprechenden Zeilennummern angegeben:

300/450 UP »Wochentag«
500/650 UP »Differenz«
800/840 UP »Tagesnummer«
900/1060 UP »Datum«
700/770 UP »Eingabe Datum«

Vom Hauptprogramm (Zeilen 10 bis 240) wird je nach der aus dem Menü ausgewählten Funktion das Unterprogramm »Wochentag«, das Unterprogramm »Differenz« oder das Unterprogramm »Datum« aufgerufen. Zur Bearbeitung der geforderten Funktionen ist in jedem dieser Unterprogramme die Eingabe mindestens eines Datums und die Berechnung der lfd. Nr. (Tagesnummer) des betreffenden Datums erforderlich. Aus diesem Grund wurden diese Berechnungen ebenfalls als Unterprogramme (UP »Eingabe Datum« und UP »Tagesnummer«) geschrieben. Auf die letzten beiden Unterprogramme wird aus den genannten Gründen von den ersten drei übergeordneten Unterprogrammen zugegriffen. Da BASIC-Programme sich im wesentlichen selbst kommentieren, soll die Erläuterung knapp gehalten werden.
Durch das Hauptprogramm wird mit einer Reihe von PRINT-Anweisungen das Menü auf dem Bildschirm erstellt

```
LI
10    REM  ::::::: KALENDER FUNKTIONEN ::::::::::::::::::
20    REM
30    REM   4. 3.1985
40    REM
50    CRTCL
60    PRINT "**********  KALENDER FUNKTIONEN  **********"
70    PRINT
80    PRINT " <1>    BESTIMMUNG DES WOCHENTAGES"
90    PRINT " <2>    BESTIMMUNG DER TAGESDIFFERENZ"
100   PRINT "        ZWISCHEN ZWEI DATEN"
110   PRINT " <3>    BESTIMMUNG EINES DATUMS AUS"
115   PRINT "        AUS DATUM UND TAGESDIFFERENZ"
116   PRINT
120   PRINT "TASTEN SIE DIE No. DER GEWUENSCHTEN"
130   PRINT "ROUTINE [1 oder 2 oder 3] EIN !"
131   PRINT
132   PRINT "JEDE EINGABE MIT [ENTER] ABSCHLIESSEN !"
133   PRINT
140   INPUT R
150   IF R=1 OR R=2 OR R=3 GOTO 180
160   PRINT CHR$(7); !! NUR EINGABE VON [1] [2] [3] MOEGLICH !!' GOTO 140
180   IF R=1 GOSUB 300
190   IF R=2 GOSUB 500
195   IF R=3 GOSUB 900
200   PRINT
210   PRINT "WIEDERHOLEN? [J/N]";
220   INPUT A$
230   IF A$="J" GOTO 40
240   STOP
299   REM  ::::::::: WOCHENTAG ::::::::::::
300   PRINT
310   PRINT "DATUM :"
320   GOSUB 700
325   PRINT
330   PRINT T;TAB(1)".";M;TAB(3)".";J;
340   GOSUB 800
350   LET L=(N+5)/7
360   LET WOT=(L-INT(L))*7
370   LET WOT=INT(WOT+.5)
380   IF WOT=0 PRINT "  SONNTAG"
390   IF WOT=1 PRINT "  MONTAG"
400   IF WOT=2 PRINT "  DIENSTAG"
410   IF WOT=3 PRINT "  MITTWOCH"
420   IF WOT=4 PRINT "  DONNERSTAG"
430   IF WOT=5 PRINT "  FREITAG"
440   IF WOT=6 PRINT "  SAMSTAG"
450   RETURN
499   REM  :::::::::: DIFFERENZ ::::::::::::
500   PRINT
505   PRINT "1. DATUM :"
510   GOSUB 700
515   LET T[1]=T LET M[1]=M LET J[1]=J
520   GOSUB 800
530   LET NN=N
540   PRINT
545   PRINT "2. DATUM :"
550   GOSUB 700
555   LET T[2]=T LET M[2]=M LET J[2]=J
560   GOSUB 800
570   LET DIFF=N-NN
580   PRINT
590   PRINT "1. DATUM :  ";T[1];TAB(14)".";M[1];TAB(16)".";J[1]
```

```
600  PRINT "2. DATUM :  ";T[2];TAB(14)".";M[2];TAB(16)".";J[2]
610  PRINT
620  PRINT "DIE DIFFERENZ ZWISCHEN DEN DATEN BETRAEGT"
630  PRINT DIFF;"TAGE"
640  PRINT
650  RETURN
699  REM  ::::::::: EINGABE DATUM  :::::::::
700  INPUT "TAG  "T
710  IF T<1 OR T>31 PRINT CHAR(7); GOTO 700
720  INPUT "MONAT"M
730  IF M<1 OR M>12 PRINT CHAR(7); GOTO 720
740  INPUT "JAHR "J
750  IF J<1598 AND J>100 OR J<0 PRINT CHAR(7); GOTO 740
760  IF J>0 AND J<100 LET J=J+1900
770  RETURN
799  REM  ::::::::::  TAGESNUMMER  :::::::::
800  LET M=M+1
810  IF M >= 4 GOTO 830
820  LET M=M+12 LET J=J-1
830  LET N=INT(J*365.25)+INT(M*30.6001)+T
840  RETURN
899  REM  ::::::::::  DATUM  ::::::::::::::::::
900  PRINT
910  PRINT "DATUM :"
920  GOSUB 700
925  PRINT
930  PRINT "AUSGANGSDATUM :  ";T;TAB(18)".";M;TAB(20)".";J
940  PRINT
950  INPUT "TAGESDIFFERENZ"DIFF
960  GOSUB 800
970  LET N=N+DIFF
980  LET J=INT((N-122.1)/365.25)
990  LET M=INT((N-INT(J*365.25))/30.6001)
1000 LET T=N-INT(J*365.25)-INT(M*30.6001)
1010 IF M >= 14 LET M=M-13
1020 ELSE  LET M=M-1
1030 IF M <= 2 LET J=J+1
1040 PRINT
1050 PRINT "ZIELDATUM :       ";T;TAB(18)".";M;TAB(20)".";J
1060 RETURN

READY
```

Bild 4. Programmliste des Programms »Kalenderfunktionen«

und die gewünschte Programmfunktion abgefragt. Die Eingabe wird auf ihre Gültigkeit hin überprüft. Bei ungültiger Eingabe erfolgt ein akustisches Signal, verbunden mit einer hinweisenden Ausschrift auf dem Bildschirm und einer erneuten Eingabeaufforderung. Entsprechend der gewählten Programmfunktion erfolgt der Unterprogrammaufruf. Nach Abarbeitung der betreffenden Unterprogramme wird nach einer eventuellen Wiederholung des Programms gefragt. Wird diese Wiederholung gewünscht, erfolgt die erneute Darstellung des Menüs, anderenfalls ein Stop der Programmabarbeitung. Im UP »Wochentag« wird der zu einem Datum gehörende Wochentag ermittelt. Hierzu ist die Eingabe des betreffenden Datums über das UP »Eingabe Datum« und die Berechnung der lfd. Nr. über das UP »Tagesnummer« erforderlich. Die Wochentagsbestimmung erfolgt wieder im UP »Wochentag« (Zeilen 350 bis 440). Durch das UP »Differenz« werden zwei Daten benötigt, die durch zweimaligen Aufruf

des UP »Eingabe Datum« bereitgestellt werden. Nach Bestimmung der zugehörigen lfd. Nr. durch zweimaligen Aufruf des UP »Tagesnummer« kann die Differenz gebildet werden. Beim UP »Datum« ist über den Aufruf des UP »Eingabe Datum« nur das Ausgangsdatum bereitzustellen. Aus der eingegebenen Tagesdifferenz und der durch das UP »Tagesnummer« berechneten lfd. Nr. kann die lfd. Nr. des Zieldatums und damit dieses selbst berechnet werden. Um in dem viel genutzten UP »Eingabe Datum« fehlerhafte Eingaben zu vermeiden, wurde auch hier ein Plausibilitätstest eingefügt. Im Falle fehlerhafter Eingaben erfolgt wieder ein akustisches Signal und eine erneute Eingabeaufforderung.
Auf Grund der Unterschiedlichkeit der verbreiteten BASIC-Dialekte wird der eine oder andere Leser unbekannte Anweisungen in der angegebenen Programmliste vorfinden. Eine Einheitlichkeit wird sich aus dem genannten Grund im allgemeinen nicht erreichen lassen. Bei der Programmiersprache BASIC ist das aber nach Meinung des Verfassers kein sonderliches Problem. Einige Korrespondenzen zwischen der hier verwendeten Version und dem BASIC des Kleincomputers KC 85/1 sollen dies verdeutlichen:

```
 50  CRTCL CLS Bildschirm löschen
160  PRINT CHAR(7) BEEP akustisches Signal
1010 IF...LET... IF...THEN...: ELSE... IF-Anw.
1020 ELSE LET... frei
```

Zu anderen Computern wird es andere Korrespondenzen geben. Im allgemeinen lassen sich jedoch immer mit der Programmliste und den dargestellten Bildschirminhalten auf der Grundlage eines bestimmten Sprachumfangs Anpassungen erreichen.

**Literatur**

[1] Kleine Enzyklopädie Natur. – 14. Aufl. – Leipzig: Verl. Enzyklopädie, 1963. – Abschn. Zeit

Autor:
*Dr.-Ing. Claus Kühnel*
Wissenschaftlicher Mitarbeiter im Zentralinstitut des Sportmedizinischen Dienstes Kreischa

## Rechentechnische Begriffe für den Laien erklärt

**RWM** Read-Write-Memory, Lese-Schreib-Speicher mit seriellem (seltener) oder parallelem (bzw. wahlfreiem) Zugriff zur gespeicherten Information; technische Realisierung meist als RAM.

**RAM** Random-Access-Memory, RWM mit wahlfreiem Zugriff zur örtlich geordneten Speicherzelle, d.h., jede Speicherzelle ist einzeln adressierbar. Als Arbeitsspeicher eingesetzt. Bei Ausfall der Betriebsspannung geht gespeicherte Information verloren, sie muß neu eingeschrieben werden.

**SRAM** Static RAM, RAM mit statischen Speicherelementen. Verlust der gespeicherten Information kann ggf. durch Bereitstellen einer zusätzlichen „Schlafspannung" (etwa 2 V) verhindert werden; Lesen und Schreiben im „Schlafzustand" aber nicht mehr möglich. Schaltkreistypen: U 202 D (2102): 1 K × 1 bit; U 214 D (2114), U 224 D: 1 K × 4 bit; U 6516: 2 K × 8 bit; 6564: 8 K × 8 bit.

# Simulation auf Mikrorechnern: Spiele und Experimente (Teil 1)

Vor etwa vierzig Jahren, als die Computer noch in ihren Kinderschuhen steckten, begann die Geschichte der Simulation auf Rechnern. Simulation ist die künstliche Nachahmung realer Vorgänge oder Prozesse. Ein Simulationsmodell oder Simulator ist ein künstlich erzeugtes System, in dem bestimmte Vorgänge oder Prozesse ablaufen. In diesem Sinne sind

- eine Spielzeugeisenbahn,
- ein Fahr- oder Flugsimulator,
- ein Videospiel oder
- ein Programm zur Nachbildung eines Produktionsprozesses

Simulationsmodelle. Jedes von ihnen ist künstliche Nachbildung eines existierenden oder nur ausgedachten Teils der Realität.
Sie unterscheiden sich

- in den nachgebildeten Gegenständen (Eisenbahn, Fahrzeug, Spielfeld oder Produktionsprozeß),
- in den Methoden der Nachbildung und
- in den Zielen der Benutzung (Spiel, Training oder Erkenntnisgewinn durch Experimente).

Traditionelle Simulationsmodelle, die einen Prozeß stofflich oder gegenständlich nachbilden, sind zwar anschaulich und vertrauenswürdig, erfordern aber meist einen hohen Realisierungsaufwand.

Ein Simulationsmodell, das einen Computer nutzt, wird in Gestalt eines Programms realisiert und heißt deshalb auch **Simulationsprogramm.** Die Elemente des realen Prozesses werden durch Daten im Operativspeicher des Rechners nachgebildet. Algorithmen, die die Änderung dieser Daten in einer Reihenfolge vornehmen, die dem realen Prozeßablauf gut entspricht, bilden den Hauptinhalt eines Simulationsprogramms.
Als Programmiersprachen für diese Simulationsprogramme kann man Maschinen- und Assemblersprachen, universelle Programmiersprachen (BASIC, PASCAL, FORTRAN, SIMULA und andere) oder Simulationssprachen (GPSS, SIMSCRIPT, SIMDIS) benutzen. Die Anwendung von Maschinen- und Assemblersprachen verursacht einen hohen Zeitaufwand für die Programmentwicklung. Solche Programme sind schwer lesbar, und ihre Änderung zur Anpassung an neue Aufgabenstellungen ist mühsam und fehleranfällig. Da unter den universellen Programmiersprachen BASIC die für Mikrorechner am weitesten verbreitete Sprache ist, soll sie im Rahmen dieses Beitrags die Basis aller vorzustellenden Programme bilden.
Viele Simulationsprogramme berücksichtigen den Umstand, daß die nachzubildenden realen Prozesse von zu-

fälligen Einflüssen abhängen. Deshalb wird sich der Teil 1 mit Methoden zur Erzeugung von Zufallszahlenfolgen und ihrer Anpassung an konkrete Eigenschaften realer Zufallsgrößen befassen.

In späteren Heften sollen folgen

Teil 2 über Simulationsmodelle von Bedienungs- und ähnlichen Prozessen und

Teil 3 über die graphische Bildschirmausgabe simulierter Prozesse in festen oder beweglichen Bildern.

## Zufallszahlenerzeugung und -transformation

Die meisten Simulationsmodelle arbeiten mit Zufallszahlen. Unter einer Zufallszahlenfolge versteht man eine Folge von Realisierungen einer Zufallsgröße. Als Beispiele solcher Folgen kann man eine Liste von 100 Würfelergebnissen, eine Folge von 6 beim Lottospiel »6 aus 49« gezogenen Zahlen oder eine Folge von 50 »Pausenzeiten« zwischen dem Eintreffen je zweier Fahrzeuge an einer Straßenverkehrskreuzung betrachten.

Um ein **Simulationsprogramm** mit Zufallszahlen zu versorgen, kann man prinzipiell folgende Wege beschreiten:

a) Erzeugung einer Zufallszahlenfolge außerhalb des Rechners und Speicherung auf einem maschinell lesbaren Datenträger,

b) Anbau eines als »Zufallszahlengenerator« wirkenden Spezialgerätes an den Rechner und

c) Erzeugung von »Pseudozufallszahlen« durch Programme.

In der Praxis hat man schon mit jedem dieser drei Wege Erfahrungen gesammelt. Die größte mit Hilfe eines »Elektronischen Roulette« erzeugte Zufallszahlentafel wurde nach ihrer Erzeugung im Jahr 1947 von der Firma RAND Corporation 1955 publiziert und enthält eine Million zufälliger Dezimalziffern. Aber derartige Zufallszahlentafeln haben ihre Nachteile: Im Operativspeicher nehmen sie zu viel Platz in Anspruch und auf peripheren Speichern dauert es zu lange, bis eine Zahl gefunden und gelesen wird. Auch direkt an den Rechner gekoppelte Zufallszahlengeneratoren bringen ihrem Nutzer Probleme: Ihre Funktionssicherheit muß überwacht werden, und ihre Resultate sind oft nicht reproduzierbar. Reproduzierbare Resultate braucht man aber besonders beim Programmtest.

In der Praxis der Simulation dominieren heute die programmierten Zufallszahlengeneratoren. Genau betrachtet, erzeugen sie keine echten Zufallszahlen, sondern reproduzierbare Zahlenfolgen, die nur »so aussehen«, als seien sie zufällig. Deshalb nennt man diese programmierten Generatoren »Pseudozufallszahlengeneratoren« und die von ihnen erzeugten Folgen »Pseudozufallszahlenfolgen«. Da im folgenden aber nur derartige Generatoren und Folgen benutzt werden, wird der Zusatz »Pseudo« der Einfachheit halber weggelassen.

In **Simulationsmodellen** benötigt man Zufallszahlenfolgen mit unterschiedlichen Eigenschaften. Resultate eines Würfels nehmen z.B. jeden der Werte 1, 2, 3, 4, 5 und 6 mit gleicher Wahrscheinlichkeit (1/6) an. Zeitabstände zwischen zwei Fahrzeugen können dagegen auch beliebige nichtganzzahlige nichtnegative Werte annehmen. Dabei kann die Wahrscheinlichkeit dafür, daß dieser Zeitabstand z.B. im Intervall von 5 bis 10 Sekunden liegt, größer sein als die Wahrscheinlichkeit für das Intervall von 10 bis 15 Sekunden. Um Zufallszahlenfolgen mit bestimmten Eigenschaften zu erzeugen, hat sich der folgende Verfahrensweg als vorteilhaft erwiesen:

Schritt 1

Erzeugung von Zufallszahlen, die im Intervall (0, 1) gleichmäßig verteilt sind,

Schritt 2

Transformation der im Schritt 1 erzeugten Zahlen entsprechend der gegebenen Wahrscheinlichkeitsverteilung.

Diesem Weg folgend, sollen Methoden und Algorithmen zur Erzeugung von Zufallszahlenfolgen vorgestellt werden, deren Elemente im Intervall (0, 1) liegen, Verfahren zu deren Transformation schließen sich an. Im Rahmen dieses Beitrags ist es nicht möglich, wahrscheinlichkeitstheoretische oder mathematisch-statistische Begriffe zu erläutern. Es wird versucht, mit möglichst wenigen Grundbegriffen auszukommen. Leser, die Schwierigkeiten beim Verstehen wahrscheinlichkeitstheoretischer oder statistischer Begriffe und Zusammenhänge bemerken, müssen auf Literatur [2, 3, 4] verwiesen werden, die für ein Selbststudium gut geeignet ist.

## 1. Standardzufallszahlengeneratoren

Generatoren, die im Intervall (0, 1) gleichverteilte Zufallszahlenfolgen erzeugen, sollen Standardzufallszahlengeneratoren genannt werden. Viele BASIC-Versionen enthalten einen derartigen Generator in Gestalt einer Standardfunktion. Sie heißt oft RND oder RN. Das ist eine Abkürzung des englischen Wortes random (Zufall). Meist ist es auch möglich, einen Anfangswert als Funktionsparameter vorzugeben. Mit unterschiedlichen Anfangswerten lassen sich dann unterschiedliche Folgen erzeugen.

### *Nutzung von BASIC-Zufallszahlengeneratoren*

Bevor gezeigt wird, wie man selbst Standardgeneratoren programmieren kann, sollen zwei Programmbeispiele die Arbeit mit einem derartigen Generator demonstrieren. Der für diese Programmbeispiele benutzte Zufallszahlengenerator erhält durch den BASIC-Befehl RANDOMIZE einen Anfangswert. RANDOMIZE erfragt diesen Wert vom Nutzer des Programms.
Im Programmbeispiel 1 wird eine Zufallszahlenfolge erzeugt. Die Länge dieser Zahlenfolge bestimmt der Nutzer. Das Programm fragt ihn nach der Anzahl der zu erzeugenden Zufallszahlen. Die erzeugten Zufallszahlen werden nun

- addiert und durch ihre Anzahl dividiert, um den Mittelwert zu berechnen, und
- in 10 Häufigkeitsklassen ($0 \cdots 0{,}1$; $0{,}1 \cdots 0{,}2$; $\cdots$; $0{,}9 \cdots 1$) eingeordnet. Für jede dieser Klassen werden die Anzahlen der in diese Klasse fallenden Zufallszahlen (die relative und die absolute Häufigkeit) berechnet und ausgegeben.

### Programmbeispiel 1

```
10 REM  Haeufigkeitstafel fuer einen
20 REM  Standardzufallsgenerator
30 REM  ----------------------------
40 REM
50 DIM HABS(10): RANDOMIZE: INPUT "n"; N
60 FOR I=1 TO N: GOSUB 160
70 FOR J=1 TO 10:
80 IF ZZ(J*.1 THEN H(J)=H(J)+1: GOTO 100
90 NEXT J
100 S = S + ZZ:   NEXT I
102 PRINT "Anzahl der Versuche", N
104 PRINT "Durchschnittswert:", S/N
110 PRINT "Obere Inter-  Absolute      Relative"
120 PRINT "vallgrenze    Haeufigkeit   Haeufigkeit"
130 FOR J=1 TO 10: PRINT J/10, H(J),
140 PRINT H(J)/N: NEXT
150 END
160 ZZ = RND
170 RETURN
```

Bild 1 zeigt das Resultat eines Programmlaufs mit $N = 100$. Dem bisher

Bild 1. Häufigkeitstafel für 1000 Zufallszahlen des BASIC-Generators

```
run
Random number seed (-32768 to 32767)? 333
n? 1000
Anzahl der Versuche                1000
Durchschnittswert:                 .49318

Obere Inter-  Absolute      Relative
vallgrenze    Haeufigkeit   Haeufigkeit
   1.           115           .115
   2            105           .105
   3            9%            .09%
   4            103           .103
   5            82            .082
  .6            93            .093
  .7            101           .101
  .8            107           .107
  .9            104           .104
   1            '94           .094
Ok
```

in der Arbeit mit Zufallszahlen ungeübten Leser wird empfohlen, dieses Programm in seinen Rechner einzugeben und einige Experimente damit zu machen. Dabei kann beobachtet werden, daß die 10 Häufigkeitsklassen unterschiedlich belegt sind und daß diese Belegung von der Wahl der Startzahl abhängt. Weiterhin beobachtet man unterschiedliche Mittelwerte. Es entstehen die Fragen,

- ob der benutzte Zufallszahlengenerator tatsächlich gleichverteilte Zahlenfolgen erzeugt und ob damit tatsächlich jedes der 10 gleichlangen Intervalle dieselbe Chance für einen »Treffer« besitzt oder
- ob man sich die auftretende Abweichung des Mittelwertes vom Erwartungswert 0,5 einer im Intervall (0, 1) gleichverteilten Zufallsgröße durch die Wirkung des Zufalls erklären kann oder ob man vermuten muß, daß die erzeugte Zahlenfolge einen anderen Erwartungswert besitzt.

Diese Fragen betreffen die Qualität der erzeugten Zufallszahlenfolge. Zufallszahlenfolgen schlechter Qualität können zu unbrauchbaren oder irreführenden Resultaten führen. Später wird auf derartige Qualitätsprobleme eingegangen.

Für ein zweites Programmbeispiel soll die folgende Aufgabe dienen: Bekanntlich sind nicht alle Integrale auf analytischem Wege lösbar, und es gibt Funktionen, deren Integral nur mit Näherungsmethoden bestimmt werden kann. Ein für einfache Integrale nicht besonders effektives, aber leicht zu programmierendes Verfahren zur näherungsweisen Integration ist die Monte-Carlo-Methode: Man bildet ein Rechteck, das von der $x$-Achse, den Integrationsgrenzen und einer oberen Schranke der zu integrierenden Funktion beschränkt wird (s. Bild 2). Nun »streut« man gleichverteilte Punkte in dieses Rechteck und zählt die Anzahl $n$ aller erzeugten Punkte sowie die Anzahl $k$ der Punkte unterhalb der zu integrierenden Funktion. Dann ist $k/n$ ein Näherungswert für das zu berechnende Integral.

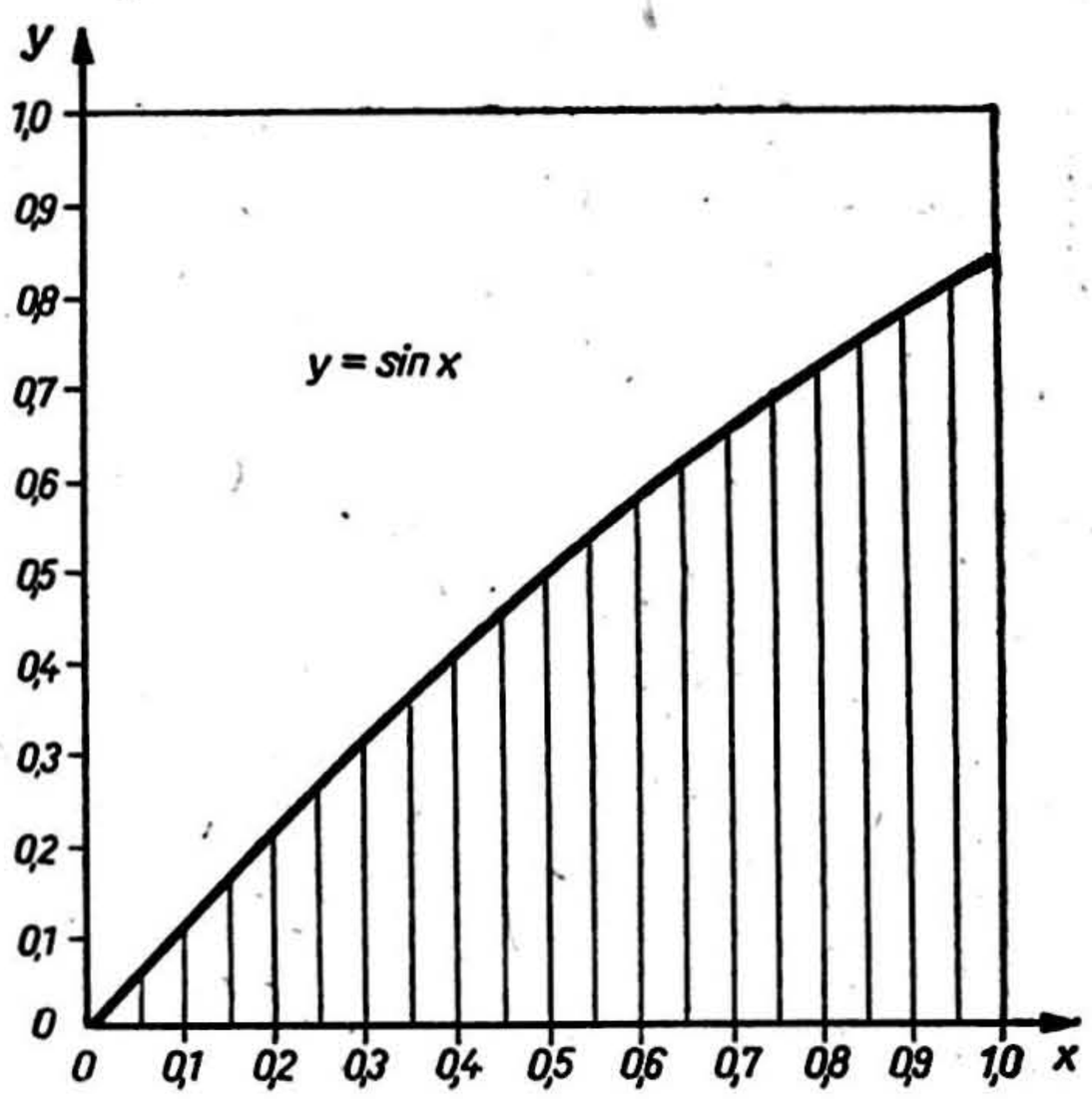



Bild 2. Bestimmung eines Flächeninhaltes mit Hilfe der Monte-Carlo-Methode

### Programmbeispiel 2

In diesem Programmbeispiel wird ein Näherungswert des Integrals der Sinusfunktion berechnet. Für diese Funktion findet man das gesuchte bestimmte Intervall in den Grenzen [0, 1] natür-

```
10 REM  Naeherungsweise Berechnung
20 REM  eines bestimmten Integrals
30 REM  mit der Monte-Carlo-Methode
40 REM  ---------------------------
50 REM
60 RANDOMIZE
70 INPUT "Anzahl zufaelliger Punkte"; N
80 FOR I = 1 TO N
90 X = RND:  Y = RND
100 IF Y < SIN(X) THEN  K = K+1
110 NEXT
120 PRINT "Naeherungswert fuer n="; N;
130 PRINT "Versuche:"; K/N
140 PRINT "Exakter Wert:          ";
150 PRINT 1 - COS(1)
160 END
```

lich viel leichter und genauer in jeder Tabelle der Winkelfunktionen: Es hat den Wert 1 — cos 1. Aber die Sinus-Funktion ist im Programm leicht durch eine andere Funktion ersetzbar, deren Integral schwieriger bestimmbar ist. Sie können mit dem Programm experimentieren und zum Beispiel die Funktion exp (— $x \times x/2$) einsetzen!

*Der Quadratmittengenerator*

Nach den ersten Versuchen mit einem BASIC-Standardgenerator soll nun gezeigt werden, wie man eigene Zufallszahlengeneratoren programmieren kann. Versuche zur Entwicklung geeigneter Algorithmen und Programme lassen sich bis in die Pionierzeit der Computerentwicklung zurückverfolgen. Alle diese Versuche haben dasselbe Ziel: Es sollen Zahlenfolgen erzeugt werden, denen keine Erzeugungsregel mehr anzusehen ist. Sie sollen gleiche statistische Eigenschaften besitzen, wie »echte«, also zum Beispiel ausgeloste Zahlenfolgen.
Einer der historisch ersten programmierten Zufallszahlengeneratoren stammt von J. VON NEUMANN, einem der Pioniere der Computerentwicklung, und wurde schon 1949 auf einem Symposium über die Monte-Carlo-Methode unter der Bezeichnung »Quadratmittenverfahren« vorgestellt:
Man nehme eine vierstellige ganze Zahl und quadriere sie. Die mittleren vier Stellen des achtstelligen (nötigenfalls links mit Nullen aufgefüllten) Resultats bilden die nächste Zahl. Sie wird wieder quadriert usw.

## Programmbeispiel 3

```
5 DIM P(5):  IP = 1
10 INPUT "startzahl"; ZZ: P(IP)=ZZ
20 GOSUB 1000: PRINT ZZ;
25 FOR J=1 TO 5: IF P(J)=ZZ THEN STOP
26 NEXT
30 IP = IP MOD 5 + 1: P(IP)=ZZ
50 GOTO 20
1000 REM Quadratmittenverfahren zur
1010 REM Zufallszahlenerzeugung
1030 REM ---------------------------
1040 ZH = ZZ * ZZ
1050 ZZ =INT(ZH/100)-INT(ZH/1E+06)*10000
1060 RETURN
```

In Bild 3 wird das Resultat der Abarbeitung dieses Programms mit der Startzahl 567 gezeigt: Nach einer zufällig erscheinenden Zahlenfolge wird relativ schnell eine Zahl erreicht, die bereits an früherer Stelle erzeugt worden war. Von da an wird die erzeugte Folge zyklisch. Das Quadratmittenverfahren erzeugt Zahlenfolgen, die nach einem mehr oder weniger langen »zyklusfreien Anfangsstück« eine unbegrenzte Folge kurzer, übereinstimmender Teilfolgen oder Zyklen erzeugen. Das oben dargestellte Programm

```
run
startzahl? 567
 3214  3297  8702  7248  5335  4622
 3628  1623  6341  2082  3347  2024
 965  9312  7133  8796  3696  6604
 6128  5523  5035  3512  3341  1622
 6308  7908  5364  7724  6601  5732
 8558  2393  7264  7656  6143  7364
 2284  2166  6915  8172  7815  7842
 5505  3050  3025  1506  2681  3234
 3269  6863  1007  140  196  384  1474
 1726  9790  8441  2504  2700  2900
 4100  8100  6100  2100  4100
Break in 25
Ok
```

**Bild 3. Vom Quadratmittengenerator mit der Startzahl 567 erzeugte Zufallszahlenfolge**

enthält einen Abschnitt, der nach derartigen Zyklen sucht und bei ihrem Auftreten die Abarbeitung unterbricht. Wie der Leser beim Experimentieren mit dem Programm bemerkt, werden damit nur Zyklen einer maximalen Länge 5 gefunden. Aber es wird nicht gelingen, eine Startzahl so zu wählen, daß ein größerer Zyklus auftritt. Dem Leser wird vorgeschlagen, dieses Programm auch einmal mit den Startzahlen 2500, 7600 oder mit der »magischen Zahl« 3792 zu erproben! (Die Zahl 3792 hat die Primfaktorenzerlegung $3972 = 3 * 79 * 2 ** 4$). Das Quadratmittenverfahren eignet sich also nur dann als Zufallszahlengenerator, wenn man mit relativ kurzen Zufallszahlenfolgen auskommt. Als verwendbar gelten die Zahlen des zyklusfreien Anfangsstückes sowie die des »nullten« Zyklus, das ist die wiederkehrende Folge bei ihrem ersten Auftreten. Es war übrigens von vornherein damit zu rechnen, daß das Quadratmittenverfahren Folgen erzeugt, die in Zyklen einmünden: Es arbeitet nach der rekursiven Erzeugungsvorschrift

$$x\,(k+1) = R\,(x\,(k)).$$

Die $(k+1)$-te Zahl wird mittels einer Rechenvorschrift $R$ aus der $k$-ten Zahl erzeugt. Da es im obigen Beispiel aber nur 10000 verschiedene Zahlen geben kann (denn alle erzeugten Zahlen sind vierstellig), muß spätestens nach 10001 Schritten eine Zahl zum zweiten Mal erzeugt werden. Nach dieser zum zweiten Mal erzeugten Zahl stimmen aber auch alle Nachfolger überein. Denn es wird ja immer dieselbe Rechenvorschrift zur Erzeugung des Nachfolgers angewandt.

*Additive und multiplikative Generatoren*

Alle praktisch bedeutsamen, zur Zufallszahlenerzeugung benutzten Algorithmen arbeiten wie das Quadratmittenverfahren mit einer rekursiven Erzeugungsvorschrift. Eine allgemeinere Form dafür lautet

$$x\,(k+1) = R\,(x\,(k), x\,(k-1), x\,(k-2), \ldots, x\,(k-r+1)).$$

In die Erzeugung der $(k+1)$-ten Zahl gehen also die $r$ zuvor erzeugten Zahlen ein. Wichtige Spezialfälle dieser allgemeinen Formel für $r = 2$ und $r = 1$ stellen additive und multiplikative Kongruentverfahren dar:

Additiver Generator:

$$x\,(k+1) = x\,(k) + x\,(k-1) \pmod{M}$$

Multiplikativer Generator:

$$x\,(k+1) = a * x\,(k) \pmod{M}$$

Um zu sichern, daß die erzeugten Zahlen zwischen Null und einem Maximalwert $M$ liegen, wird »modulo $M$« gerechnet: Wenn das Resultat der Rechnung größer oder gleich $M$ wird, nimmt man den Rest aus seiner Division durch $M$. Die Rechnung erfolgt immer mit ganzen Zahlen. Zahlen im Intervall (0, 1) gewinnt man durch nachträgliche Division durch $M - 1$. Das folgende Programmbeispiel bietet dem Leser die Möglichkeit, selbst erste Erfahrungen im Umgang mit additiven und multiplikativen Zufallszahlengeneratoren zu sammeln.

**Programmbeispiel 4**

Erfahrungsgemäße und auch theoretisch begründete Kenntnisse über additive und multiplikative Zufallszahlengeneratoren findet der Leser in der Fachliteratur über Zufallszahlen, z. B. [5]. Hier sollen nur wenige Fakten über Eigenschaften dieser Generatoren wiedergegeben werden.

(1) Alle additiven und multiplikativen Generatoren sind zyklisch. Die erzeugten Folgen können mit einem

```
10 REM  Demonstrationsbeispiele fuer
20 REM  rekursive ZZ-Generatoren
30 REM  ------------------------------
40 REM  (1) Additiver Generator:
50 REM      I(n+1) = I(n) + I(n-1)
60 PRINT "Eingabe zweier positiver"
70 PRINT "ganzer Zahlen (Startzahlen):"
80 INPUT "1. Startzahl"; I1
90 INPUT "2. Startzahl"; I2
100 INPUT "Modul"; MM
110 INPUT "Umfang der Ausgabefolge"; NF
120 FOR I = 1 TO NF: IH = I2:
130     I2 = I1 + I2: I1 = IH
140     I2 = I2 - INT(I2/MM)*MM
150   PRINT I2; I2/MM;: NEXT
160 PRINT
170 REM (2) Multiplikativer Generator:
180 REM     IZ(n+1) = AA * IZ(n) mod M
190 PRINT "Startzahl fuer den multi";
200 INPUT "plikativen Generator"; I1
220 INPUT "Multiplikator"; AA
230 INPUT "Modul"; MM
240 INPUT "Umfang der Ausgabefolge"; NF
250 IF AA>MM THEN PRINT "Eingabefehler:    AA > MM"
260 I1 = I1 - INT(I1/MM)*MM
270 FOR I = 1 TO NF: I1 = I1 * AA
280     I1 = I1 - INT(I1/MM) * MM
290     PRINT I1; (I1-1)/(MM-2);: NEXT
300 END
```

zyklenfreien Anfangsstück beginnen. Darauf folgt ein »nullter Zyklus«. Multiplikative Generatoren treten dann in ihren ersten Zyklus ein, wenn eine bereits erzeugte Zahl zum zweiten Mal auftritt.

Additive Generatoren, die jeweils zwei vorher erzeugte Zahlen für die Berechnung der nächsten Zahl benutzen, beginnen ihren ersten Zyklus, wenn ein Paar benachbarter Zahlen zum zweiten Mal auftritt.

(2) Die maximale Zykluslänge oder Periode eines multiplikativen Generators mit dem Modul $M = 2 ** n$ ist gleich $2 ** (n - 2)$. Sie wird erreicht, wenn die Startzahl ungerade und der Faktor $AA$, mit dem jedesmal multipliziert wird, gleich 3 oder 5 (mod 8) ist.

Zum Abschluß der Beschreibung von Zufallszahlengeneratoren soll im folgenden Programmbeispiel ein Generator angeführt werden, der mit nichtganzzahligen Werten operiert. Auch er erzeugt zyklische Zahlenfolgen. Durch geeignete Wahl von Startzahl und Multiplikator erreicht man große Zykluslängen. Sie bilden ein Qualitätsmerkmal jedes Generators. In der Praxis der Simulation sollte immer nur das zyklenfreie Anfangsstück und der nullte Zyklus verwendet werden.

**Programmbeispiel 5**

```
10 REM Multiplikativer Generator
20 REM
30 PRINT "Ungerade positive ganz-"
40 INPUT "zahlige Startzahl"; Z0
50 ZZ = Z0/2.09715E+06
60 GOSUB 120
70 K = K+1
80 IF K=10 THEN ZV = ZZ
90 IF K/100=INT(K/100) THEN PRINT ZZ;
100 IF K>10 AND ZV=ZZ THEN PRINT "Zyklusbeginn:  k="; K;
"  zz="; ZZ: STOP
110 GOTO 60
120 ZZ = ZZ * 509: ZZ = ZZ - INT(ZZ)
130 RETURN
```

Durch Experimente mit diesem Programm erkennt man, daß die Zykluslänge der erzeugten Zufallszahlenfolge dann besonders kurz wird, wenn man entgegen der Eingaberegel Potenzen von 2 als Startzahlen eingibt.

Für viele Anwendungen von Zufallszahlen, insbesondere für Spiele, ist es wünschenswert, daß die Startzahl nicht vom Benutzer »eingestellt« werden kann. Das folgende, letzte Programmbeispiel dieses Abschnittes zeigt ein

einfaches Verfahren, das dem Nutzer des Programms die Kontrolle über die Startzahl entzieht. Es benutzt die nicht in allen BASIC-Versionen verfügbare Funktion INKEY$, diese Funktion hat als Wert ein über die Tastatur des Rechners eingegebenes beliebiges Zeichen.

**Programmbeispiel 6**

```
10 REM Multiplikativer Generator
20 REM mit zufaelliger Startzahl
30 REM
40 PRINT "Beliebige Eingabe:"
50 Z0 = Z0 + 1
60 IF INKEY$ <> "" GOTO 90
70 IF Z0 > 1000 THEN Z0 = Z0-1000
80 GOTO 50
90 IF Z0/2=INT(Z0/2) THEN Z0=Z0+1
100 PRINT "z0 ="; Z0
110 ZZ = Z0/2.09715E+06
120 GOSUB 160
130 K = K+1
140 IF K<10 THEN PRINT ZZ; ELSE STOP
150 GOTO 120
160 ZZ = ZZ * 509: ZZ = ZZ - INT(ZZ)
170 RETURN
```

Damit sind nun ausreichende Voraussetzungen dafür gegeben, Standardzufallszahlenfolgen zu erzeugen. Im nächsten Abschnitt werden Methoden zur Transformation dieser, im Intervall (0, 1) gleichmäßig verteilten Zufallszahlenfolgen in Folgen mit anderen Verteilungsgesetzen vorgestellt.

## 2. Transformation von Zufallszahlen

Nachdem Möglichkeiten zur Erzeugung von gleichverteilten Zufallszahlen im Intervall (0, 1) beschrieben wurden, ergibt sich nun die Fragestellung: Wie können mit diesen Zufallszahlen die benötigten zufälligen Ereignisse im Simulationsmodell nachgebildet werden? Die mit Hilfe der Generatoren gewonnenen Zufallszahlen stellen noch nicht die möglichen Augenzahlen beim Würfeln dar. Für die Programmierung des MASTER-MIND-Spiels werden ganzzahlige Zufallszahlen im Intervall von 0 bis 9 benötigt. Zur Simulation eines Kartenspiels müssen die Karten an die Mitspieler verteilt werden. Dieses Verteilen können ganzzahlige Zufallszahlen aus dem Bereich von 1 bis 32 übernehmen. Besteht die Aufgabe, z.B. einen Schießwettbewerb von Sportschützen zu simulieren, so weiß man aus der Erfahrung, daß die Treffer von Sportschützen sich in der Mitte der Scheibe konzentrieren, während »Fahrkarten« kaum geschossen werden. Die erreichten Ringe können durch ganzzahlige Zufallszahlen aus dem Bereich von 0 bis 10 dargestellt werden, nur darf die Verteilung nicht gleichmäßig über alle Werte sein.
Neben diesen ganzzahligen Zufallszahlen, sie gehören in die Gruppe der Zufallszahlen mit diskreter Verteilung, benötigt man zur Simulation technischer Prozesse häufig Zufallszahlen mit stetiger Verteilung. Diese Zufallszahlen können in einem festgelegten Intervall jeden Wert annehmen.
Beispiele derartiger Zufallsgrößen sind

- die Fahrzeit, die Fahrzeuge für eine gegebene Strecke benötigen,
- die Dauer einer Zeitperiode, in der eine Maschine störungsfrei arbeitet,
- die Brennstoffmenge, die von einer Anlage je Zeiteinheit verbraucht wird.

Aus diesen wenigen Beispielen ist ersichtlich, daß eine Transformation der gleichverteilten Zufallszahlen in verschiedene Verteilungsformen notwendig ist. Im folgenden werden entsprechende Transformationsvorschriften vorgestellt.

*Gleichverteilte Zufallszahlen im Intervall (a, b)*

Bei der Transformation von gleichverteilten Zufallszahlen aus dem Intervall (0, 1) in das Intervall ($a$, $b$) werden hier 2 Fälle unterschieden:

(1) Die transformierten Zufallszahlen besitzen eine stetige gleichmäßige Verteilung, und
(2) die transformierten Zufallszahlen nehmen nur die ganzzahligen Werte aus diesem Intervall mit jeweils gleicher Wahrscheinlichkeit an.

Für den ersten Fall kann die Transformation nach folgender BASIC-Anweisung realisiert werden:

$ZT = A + (B - A) * ZZ$

$A$ gibt die untere Grenze, $B$ die obere Grenze des Intervalls an. $ZZ$ ist eine Zufallszahl aus dem Intervall (0, 1). Dem Leser soll es hier selbst überlassen sein, aus dieser Programmzeile ein Programm zu entwickeln und zur Wertzuweisung an die Variable $ZZ$ einen eigenen Generator oder die Funktion RND zu verwenden. Zur Erzeugung von ganzzahligen Zufallszahlen aus dem Intervall $(a, b)$ muß eine andere Vorschrift verwendet werden. Ermittelt man die Zufallszahlen nach der obigen Vorschrift für das Intervall von $a = 1$ bis $b = 6$, so kann die Zahl 6 nicht erzeugt werden, da die mittels Generatoren bereitgestellten Zufallszahlen immer kleiner als 1 sind. Diesen Umstand gilt es bei der Transformation zu berücksichtigen.
Für die Erzeugung von gleichverteilten ganzzahligen Zufallszahlen aus dem Intervall $(a, b)$ kann folgende BASIC-Anweisung verwendet werden:

$ZT = A + \mathrm{INT}((B - A + 1) * ZZ)$

Mit dem folgenden Programmbeispiel wird ein Roulettespiel simuliert. Die Randpunkte des benötigten Intervalls sind $a = 0$ und $b = 36$. Der Spieler hat in diesem Programm nur die Möglichkeit, auf gerade oder ungerade Zahlen einen beliebigen Betrag zu setzen. Ausgehend von einem Ausgangskapital und dem entsprechenden Einsatz des Spielers, wird nach jedem Spiel eine Bilanz gezogen. Zur Erzeugung der Zufallszahlen wird die Funktion RND verwendet, deren Startzahl über RANDOMIZE festgelegt wird. Damit der Spieler die Startzahl nicht beeinflussen kann, wird sie aus dem Produkt der Minuten- und Stundenangaben einer internen Uhr des Mikrorechners übernommen. Für den verwendeten Computer sind die aktuellen Zeitangaben unter den Adressen 80 und 81 gespeichert. Mit dem PEEK-Befehl wird auf diese Adresse zugegriffen.
Zur Steuerung des Cursors auf dem Bildschirm für die Resultatausgabe sind Steuerzeichen verwendet worden. Auf die Thematik zur Bereitstellung und Verwendung von Cursorsteuerzeichen wird später eingegangen werden. Dem Leser, der mit dieser Problematik noch nicht vertraut ist, wird die Verwendung des »normalen« PRINT-Befehles empfohlen.

**Programmbeispiel 7**

Während der Abarbeitung des Programms (s. S. 34 links) erhält man die in Bild 4 gezeigte Darstellung.
Dieses Programm stellt nur einen Torso für eine vollständige Simulation des Roulettespiels dar. Dem Leser ist es selbst überlassen, dieses Programm für den privaten Gebrauch zu erweitern. Als Ergänzungen sind vorzuschlagen:

- Einsatz auf alle zulässigen Zahlenkombinationen,
- Erweiterung des Spielerkreises auf eine beliebige Anzahl,
- Erfolgsstatistik für die Bank und
- graphische Darstellung des Rollens der Kugel im Roulette.

Das Programmbeispiel 8 ist den Lesern gewidmet, die ihren Computer einmal als einen Spielautomaten benutzen

```
10 REM Simulation Roulette
20 REM Zuweisungen fuer Druck
30 LF=10: FF=12: SUB=26: CAN=24:NAK=21
40 SYN=22: LZ$=CHR$(SYN)
50 OBEN$=CHR$(SUB): UNTEN$=CHR$(LF)
60 LO$=CHR$(CAN): LBS$=CHR$(FF)
70 FOR I=1 TO 40:RE$=RE$+CHR$(NAK):NEXT
80 REM Initialisierung
90 FOR J=1 TO 24: ST$=ST$+"*": NEXT
100 DIM TEXT$(2): TEXT$(0)="gerade  "
110 TEXT$(1)="ungerade"
120 INPUT "Ausgangskapital:";AK: AKK=AK
130 SZ=PEEK(80)*PEEK(81): RANDOMIZE(SZ)
140 REM Ausgabe der Textschablone
150 PRINT LBS$;UNTEN$;UNTEN$;UNTEN$;
160 FOR J=1 TO 24: MAR$=MAR$+"-":NEXT
170 PRINT MAR$
180 PRINT "Erspielte Zahl...:"
190 PRINT "Einsatz.........:"
200 PRINT "Erzielter Gewinn.:"
210 PRINT "Gesetzte Zahlen..:"
220 PRINT "Erspielte Zahlen.:"
230 PRINT "Ausgangskapital..:"
240 PRINT "Aktuelles Kapital:"
250 PRINT "Spielnummer......:"
260 REM Eingabedaten fuer das Spiel
280 PRINT LO$;LZ$;: SN=SN+1
290 INPUT "Ihr Einsatz:";ES
300 PRINT LZ$;
310 INPUT "Zahlentyp (g/u):";T$
320 PRINT LZ$
330 IF T$="G" OR T$="g" THEN TW=0
340 IF T$="U" OR T$="u" THEN TW=1
350 PRINT LO$;ST$
360 PRINT: PRINT ST$
370 PRINT OBEN$;OBEN$;
380 FOR I=1 TO 5: PRINT LZ$;
390 FOR J=1 TO 300: NEXT
400 PRINT "* ROULETTE DREHT       *"
410 PRINT OBEN$;
420 FOR J=1 TO 300: NEXT J: NEXT I
430 REM Ermittlung der Zahl
440 ZZ=RND: ZT=INT(37*ZZ)
450 REM Auswertung
460 IF ZT=0 GOTO 500
470 X=ZT MOD 2
480 IF X<>TW GOTO 500
490 EG=2*ES: AKK=AKK+EG-ES: GOTO 510
500 EG=0: AKK=AKK-ES
510 PRINT LEFT$(RE$,17);
520 PRINT "NICHT";UNTEN$;UNTEN$
530 DR$=LEFT$(RE$,18)
540 PRINT DR$;ZT: PRINT DR$;ES
550 PRINT DR$;EG: PRINT DR$;TEXT$(TW)
560 IF ZT=0 THEN PRINT DR$;"BANKGEWINN"
570 PRINT DR$;TEXT$(X):PRINT DR$;AK
580 PRINT DR$;AKK: PRINT DR$;SN
590 PRINT LZ$;
600 REM Spielabbruch
610 INPUT "Neues Spiel (j/n)";ANT$
620 IF ANT$="J" OR ANT$="j" GOTO 280
630 END
```

wollen. Mit diesem Programm wird ein Spielautomat nachgebildet, wie er des öfteren auf Rummelplätzen zu finden ist. Aus einer genügend großen Anzahl von Spielen, die der Leser ohne Geld-

```
Ihr Einsatz:? 10
Zahlentyp (g/u):? u

-----------------------
Erspielte Zahl...: 35
Einsatz.........: 10
Erzielter Gewinn.: 20
Gesetzte Zahlen..:ungerade
Erspielte Zahlen.:ungerade
Ausgangskapital..: 100
Aktuelles Kapital: 110
Spielnummer.....: 2
Neues Spiel (j/n)? j
```

**Bild 4. Bildschirmdarstellung des Roulettespiels**

einsatz spielen kann, soll er sich selbst ein Urteil darüber bilden, ob die Bezeichnung »Einarmiger Bandit« gerechtfertigt ist.

## Programmbeispiel 8

```
10 REM Einarmiger Bandit
20 REM Zuweisungen fuer Ausgabe
30 FF=12: LBS$=CHR$(FF)
40 REM Initialisierung
50 DIM Z(4): GEW=0: RANDOMIZE
60 REM Standardteil drucken
70 PRINT " S P I E L A U T O M A T "
80 PRINT "  ----------------------
90 PRINT : PRINT " Spielregeln "
100 PRINT "Zwei gleiche Ziffern :";
110 PRINT " Doppelter Einsatz "
120 PRINT "Drei gleiche Ziffern :";
130 PRINT " Vierfacher Einsatz "
140 PRINT
150 REM Eingabedaten
160 INPUT " Einsatz ==> ",EINS: SW=1
170 PRINT LBS$
180 REM Zahlen auswaehlen und ausgeben
190 FOR I=1 TO 15
200 FOR J=1 TO 3: Z(J)=INT(10*RND):NEXT
210 PRINT LBS$;
220 PRINT Z(1);Z(2);Z(3)
230 FOR K=1 TO I*I+20: NEXT K,I
240 REM Spielauswertung
250 IF Z(1)=Z(2) THEN SW=SW+1
260 IF Z(2)=Z(3) THEN SW=SW+1
270 IF Z(3)=Z(1) THEN SW=SW+1
280 ON SW GOTO 290,300,310,310
290 SG=0: GOTO 320
300 SG=EINS*2: GOTO 320
310 SG=EINS*4
320 GEW=GEW+SG-EINS
330 PRINT "Gewinn des Spieles :";SG
340 PRINT "Gesamtgewinn       :";GEW
350 INPUT "Neues Spiel (j/n) ";ANT$
360 IF ANT$="J" OR ANT$="j" THEN  160
370 END
```

*Diskrete Zufallszahlen mit empirischer Verteilung*

Anders als bei den obigen, auf Glücksspiele bezogenen Beispielen verhalten sich viele Zufallsgrößen, denen man in der Realität begegnet: die Wahrscheinlichkeit dafür, daß

- beim Schießen 0, 1, 2, . . ., 10 Ringe getroffen werden,
- das nächste, an einer Kreuzung eintreffende Fahrzeug ein PKW, LKW, Moped oder Fahrrad ist,
- der Himmel am 31. 12. 1986 bedeckt, bewölkt, heiter oder wolkenlos ist,
- während einer Schicht 0, 1, 2, 3 oder 4 Maschinen ausfallen,

sind nicht untereinander gleich. Zur Simulation derartiger Ereignisse benötigt man in der Regel statistische Angaben über Häufigkeiten ihres Auftretens.

Zum Beispiel liege folgende Statistik vor:

Anzahl der täglichen Maschinenausfälle in einem Beobachtungszeitraum von 100 Tagen

| Anzahl | absolute Häufigkeit | relative Häufigkeit | empirische Verteilungsfunktion |
|---|---|---|---|
| 0 | 6 | 0,06 | 0,06 |
| 1 | 15 | 0,15 | 0,21 |
| 2 | 24 | 0,24 | 0,45 |
| 3 | 50 | 0,50 | 0,95 |
| 4 | 4 | 0,04 | 0,99 |
| 5 | 1 | 0,01 | 1,00 |

Man kann sich die Erzeugung entsprechend verteilter Zufallszahlen an Bild 5 erklären:

Die erzeugte Zufallszahl aus (0, 1) trägt man auf der Ordinate ab (z. B. $ZZ = 0{,}62$) und bestimmt den Schnittpunkt mit der Treppenfunktion, die die empirische Verteilung darstellt (z. B. $ZT = 3$). Die größte Chance, von einer

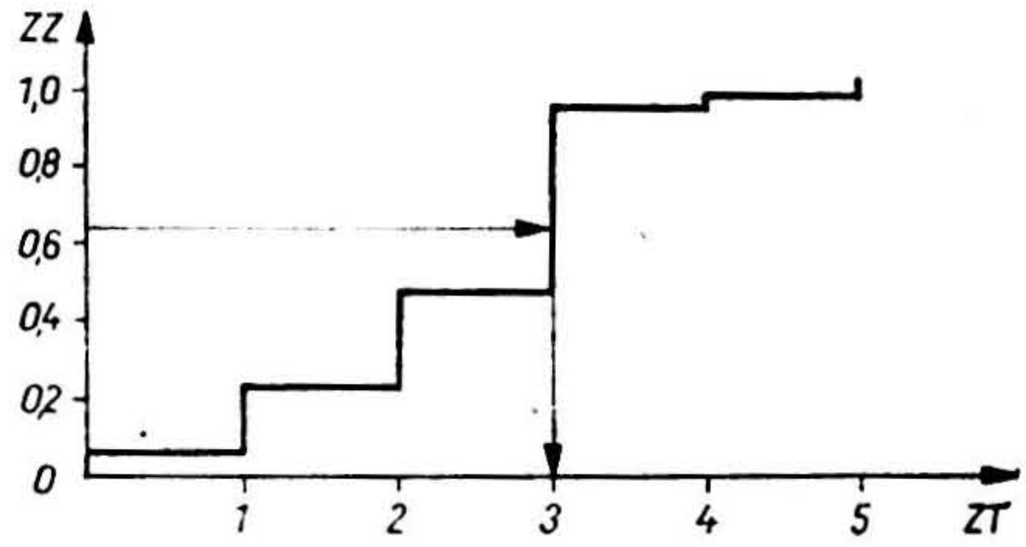



Bild 5. Erzeugung von Zufallszahlen mit diskreter Verteilung

Zufallszahl $ZZ$ getroffen zu werden, hat das Ereignis »Anzahl = 3«, das in der Realität am häufigsten beobachtet wurde. Der im folgenden vorgestellte Algorithmus realisiert diesen Verfahrensweg.

In der Praxis können diskrete Zufallsgrößen bedeutend mehr Werte annehmen, als in den obigen Beispielen. Es ergibt sich für den Nutzer von Simulationsmodellen die Frage, ob es immer wieder notwendig ist, die entsprechenden Häufigkeiten für diese Werte einzugeben, oder ob es nicht möglich ist, die Werte auf einen Datenträger abzuspeichern und diese abgespeicherten Daten zur Transformation nutzen zu können. Leider verfügen nicht alle BASIC-Versionen über die notwendigen Ein- und Ausgabebefehle.

Die folgenden BASIC-Programme verwenden die Befehle OPEN und CLOSE zum Eröffnen bzw. zum Schließen von Dateien sowie die Befehle PRINT und INPUT zum Beschreiben einer Datei bzw. zum Lesen von Daten aus einer Datei. Dem Leser, der diese Befehle nicht verwenden kann, sei empfohlen, sein Programm durch gewöhnliche Eingabebefehle mit den nötigen Werten zu versorgen.

Das erste BASIC-Programm beschreibt eine sequentielle Datei, deren Namen der Nutzer eingeben muß. Der Nutzer wird weiterhin aufgefordert, die Anzahl der Werte, die die Zufallsgröße

annehmen kann, den Wert der Zufallsgröße sowie die absolute Häufigkeit einzugeben. Über die Berechnung der relativen Häufigkeiten erfolgt die Bestimmung der Verteilungsfunktion. Die zur Transformation benötigten Daten werden auf die entsprechende Datei ausgegeben. Nach erfolgreicher Übertragung wird dem Nutzer eine entsprechende Nachricht ausgegeben.

**Programmbeispiel 9 Teil 1**

```
10 REM Transformation in
20 REM Zufallszahlen mit empirischer
30 REM diskreter Verteilung
40 REM Teil 1 - Statistische Aufberei-
50 REM tung und Ausgabe auf Datei
60 REM SN Anzahl der Werte, die die
65 REM    Zufallsgroesse annehmen kann
70 REM SA Absolute Haeufigkeit eines
80 REM    Wertes der Zufallsgroesse
90 REM SH Zwischensumme
100 REM Werte der Zufallsgroesse
110 DIM SW(30)
120 REM Relative Haeufigkeiten
130 DIM RH(30)
140 REM Werte der Verteilungsfunktion
150 DIM VE(30)
160 REM DSN$  Dateiname
170 SH=0
180 PRINT "Anzahl der Werte,  welche die"
190 INPUT "Zufallsgroesse annehmen kann:";SN
200 FOR I=1 TO SN
210 PRINT "Wert: ";I; ": ";
220 INPUT SW(I)
230 INPUT "Absolute Haeufigkeit: ";SA
240 RH(I)=SA: SH=SH+SA: NEXT
250 REM Berechnung d. relativ. Haeufigk.
260 FOR I=1 TO SN: RH(I)=RH(I)/SH: NEXT
270 REM Empirische Verteilungsfunktion
280 VE(0)=0: VE(SN)=1!
290 FOR I=1 TO SN-1:VE(I)=VE(I-1)+RH(I)
300 NEXT
310 REM Ausgabe auf Datei DSN$
320 INPUT "Ausgabe auf Datei :";DSN$
330 OPEN "O",#1,DSN$
340 PRINT# 1,SN
350 FOR I=1 TO SN
360 PRINT# 1,SW(I): NEXT I
370 FOR I=0 TO SN
380 PRINT# 1,VE(I): NEXT I
390 CLOSE #1
400 PRINT "Alle Werte geschrieben auf "
410 PRINT "Datei ";DSN$
420 END
```

Für das Beispiel »Maschinenausfälle« ergibt sich die in Bild 6 wiedergegebene Darstellung.
In dem zweiten BASIC-Programm wird der Nutzer nach dem Namen der Datei gefragt, von der die entsprechenden Daten zur Transformation gelesen werden. Die erfolgreiche Datenübertragung wird dem Nutzer mitgeteilt. Der Nutzer kann hier auf verschiedene Dateien zugreifen, in denen die erforderlichen Daten für unterschiedliche Zufallsgrößen abgespeichert sind, ohne Veränderungen am Programm vornehmen zu müssen. Für die Erzeugung der gleichverteilten Zufallszahlen wird die Funktion RND verwendet. Zur Demonstration werden in diesem Programm die Werte von 10 gleichverteilten Zufallszahlen mit ihren entsprechenden transformierten Werten ausgegeben.

```
run
Anzahl der Werte,  welche die
Zufalssgroesse annehmen kann:? 6
Wert:  1 : ? 0
Absolute Haeufigkeit: ? 6
Wert:  2 : ? 1
Absolute Haeufigkeit: ? 15
Wert:  3 : ? 2
Absolute Haeufigkeit: ? 24
Wert:  4 : ? 3
Absolute Haeufigkeit: ? 50
Wert:  5 : ? 4
Absolute Haeufigkeit: ? 4
Wert:  6 : ? 5
Absolute Haeufigkeit: ? 1
Ausgabe auf Datei :? "B:DAT1"
Alle Werte geschrieben auf
Datei B:DAT1
Ok
```

Bild 6. Eingabe der Daten zur Beschreibung einer diskreten Verteilung

**Programmbeispiel 9 Teil 2**

Bild 7 zeigt die Abarbeitung des Programms für das Beispiel »Maschinenausfälle«.

```
10 REM Transformation in
20 REM Zufallszahlen mit empirischer
30 REM diskreter Verteilung
40 REM Teil 2 - Einlesen von einer
50 REM Datei und Transformation
60 DIM SW(30): DIM VE(30)
70 INPUT "Einlesen von Datei:",DSN$
80 REM Lesen in Datei DSN$
90 OPEN "I",#1,DSN$
100 INPUT# 1,SN
110 FOR I=1 TO SN
120 INPUT# 1,SW(I): NEXT I
130 FOR I=0 TO SN
140 INPUT# 1,VE(I): NEXT I
150 CLOSE #1
160 PRINT "Alle Werte gelesen aus "
170 PRINT "Datei ";DSN$
180 REM Transformation
190 RANDOMIZE
200 FOR J=1 TO 10: ZZ=RND
210 GOSUB 250
220 PRINT "ZZ=";ZZ,"ZT=";ZT
230 NEXT
240 GOTO 310
250 REM UP DISKRET
260 FOR I=1 TO SN
270 IF ZZ > VE(I) THEN GOTO 290
280 ZT=SW(I): GOTO 300
290 NEXT
300 RETURN
310 END
```

### *Stetige Zufallszahlen mit empirischer Verteilung*

In vielen Simulationsexperimenten werden Zufallsvariablen mit stetiger Verteilung benötigt. Als praktisches Beispiel wird eine zufällige Zeitdauer betrachtet. Beobachtungen über die Dauer und deren statistische Aufbereitung werden Zeitaufnahmen genannt. Teilt man den gesamten ermittelten Wertebereich in gleich große Abschnitte oder Klassen ein und ordnet man die ermittelten Meßwerte den entsprechenden Klassen zu, so ergibt sich eine empirische Verteilung der absoluten Häufigkeiten für die jeweiligen Klassen.

```
run
Einlesen von Datei."B DAT1"
Alle Werte gelesen aus
Datei B:DAT1
Random number seed (-32768 to 32767)? 25
ZZ=  .584838    ZT= 3
ZZ=  .206153    ZT= 1
ZZ=  .682343    ZT= 3
ZZ=  .891111    ZT= 3
ZZ=  .173183    ZT= 1
ZZ=  .202896    ZT= 1
ZZ=  .633494    ZT= 3
ZZ=  .530886    ZT= 3
ZZ=  .627636    ZT= 3
ZZ=  .2808      ZT= 2
Ok
```

**Bild 7. Resultate der Transformation und 10 gleichverteilten Zufallszahlen in Zufallszahlen mit einer diskreten Verteilung**

Das BASIC-Programm Seite 38 ermöglicht die Transformation von gleichverteilten Zufallszahlen aus dem Intervall (0, 1) in Zufallszahlen mit einer empirischen stetigen Verteilung. Ausgangsgrößen sind die absoluten Häufigkeiten der Beobachtungswerte in den einzelnen Klassen. Aus diesen absoluten Häufigkeiten werden die relativen Häufigkeiten und die entsprechende Verteilungsfunktion ermittelt.

Das Programm erfragt zum Beginn die Anzahl der verwendeten Klassen, die als konstant vorausgesetzte Klassenbreite und die untere Grenze der 1. Klasse. Für jede Klasse erfolgt die Eingabe der absoluten Häufigkeiten. Im zweiten Teil des Programms wird die eigentliche Transformation durchgeführt. Die gleichverteilte Zufallszahl wird mit der Verteilungsfunktion verglichen, um die nun zufällig ausgewählte Häufigkeitsklasse anzugeben. Zur Ermittlung der stetigen Zufallsgröße aus dieser Klasse wird eine lineare Interpolation zwischen der unteren und oberen Klassengrenze durchgeführt. Auf die Ableitung der entsprechenden Transformationsvorschriften wird hier verzichtet, und der interessierte Leser sei auf [1] verwiesen. Als Zufallszahlengenerator für die gleichverteilten Zufallszahlen wird die BASIC-Funktion RND verwendet. Zur Demonstration werden 10 gleichverteilte Zufallszahlen in Zufallszahlen mit der eingegebenen empirischen Verteilung transformiert.

## Programmbeispiel 10

```
10 REM Transformation gleichverteil-
20 REM ter Zufallszahlen in
30 REM Zufallszahlen beliebiger
40 REM stetiger Verteilung
50 REM KN  Anzahl der Klassen
60 REM KB  Klassenbreite
70 REM KA  Absolute Klassenhaeufigkeit
80 REM UG  Untere Klassengrenze der
90 REM     1. Klasse
100 REM Relative Klassenhaeufigkeit
110 DIM RH(30)
120 REM Untere Klassengrenzen
130 DIM KU(30)
140 REM Verteilungsfunktion
150 DIM VE(30)
160 INPUT "Anzahl der Klassen: "; KN
170 INPUT "Klassenbreite     : "; KB
180 PRINT "Untere Grenze d. 1. Klasse:";
190 INPUT UG
200 SH=0: KU(0)=UG-KB
210 FOR I=1 TO KN
220 KU(I)=KU(I-1)+KB
230 PRINT"Absolute Haeufigkeit der Kla";
240 PRINT "sse"; I
250 PRINT " (";KU(I);" - ";KU(I)+KB;")";
260 INPUT KA: RH(I)=KA: SH=SH+KA: NEXT I
270 REM Berechn. d. relat. Haeufigkeit
280 FOR I=1 TO KN: RH(I)=RH(I)/SH: NEXT
290 REM Berechnung d. Verteiliungsfkt.
300 VE(0)=0: VE(KN)=1!
310 FOR I=1 TO KN-1: VE(I)=VE(I-1)+RH(I)
320 NEXT
330 REM Transformation
340 RANDOMIZE
350 FOR J=1 TO 10: ZZ=RND
360 GOSUB 400
370 PRINT "ZZ=";ZZ,"ZT=";ZT
380 NEXT
390 GOTO 470
400 REM UP STETIG
410 FOR I=1 TO KN
420 IF ZZ > VE(I) THEN GOTO 450
430 ZT=KU(I) + (ZZ-VE(I-1))*KB/RH(I)
440 GOTO 460
450 NEXT
460 RETURN
470 END
```

Für eine Zeitaufnahme über 3 Klassen erhält man die in Bild 8 gezeigte Darstellung.

```
run
Anzahl der Klassen: ? 3
Klassenbreite     : ? 10
Untere Grenze d  1. Klasse:? 20
Absolute Haeufigkeit der Klasse 1
 ( 20  -  30 )? 25
Absolute Haeufigkeit der Klasse 2
 ( 30  -  40 )? 48
Absolute Haeufigkeit der Klasse 3
 ( 40  -  50 )? 35
Random number seed (-32768 to 32767)? 25
ZZ=  .584838    ZT= 37.9505
ZZ=  .206153    ZT= 28.9058
ZZ=  .682343    ZT= 40.198
ZZ=  .891111    ZT= 46.64
ZZ=  .173183    ZT= 27.4815
ZZ=  .202896    ZT= 28.7651
ZZ=  .633494    ZT= 39.0453
ZZ=  .530886    ZT= 36.7366
ZZ=  .627636    ZT= 38.9135
ZZ=  .2808      ZT= 31.1097
Ok
```

Bild 8. Resultate der Transformation von 10 gleichverteilten Zufallszahlen in Zufallszahlen mit einer stetigen Verteilung

## Literatur


[1] Frank, M., Lorenz, P.: Simulation diskreter Prozesse. – Leipzig: Fachbuchverlag, 1979

[2] Maibaum, *G.*: Wahrscheinlichkeitsrechnung. – Berlin: Verlag Volk und Wissen, 1971

[3] Beyer, O.; Hackel, H.; Pieper, V.; Tiedge, J.: Wahrscheinlichkeitsrechnung und mathematische Statistik. – Leipzig: Teubner Verlag, 1976

[4] Storm, R.: Wahrscheinlichkeitsrechnung, mathematische Statistik und statistische Qualitätskontrolle. – 8. Aufl. – Leipzig: Fachbuchverlag, 1985

[5] Zieliński, R.: Erzeugung von Zufallszahlen. – Leipzig: Fachbuchverlag 1978



Autoren:
*Prof. Dr. Peter Lorenz*

*Dr. Thomas Schulze*

Technische Hochschule
„Otto von Guericke" Magdeburg
Sektion Rechentechnik
und Datenverarbeitung

# Master Mind – gegen den Rechner gespielt

Master Mind ist ein Zahlenratespiel, welches fast jeder auf seinem Rechner zu realisieren versucht. Es bietet sich dafür auch an, da es eine Zahlenmanipulation erfordert und einen relativ einfachen Auswertealgorithmus besitzt. Die Kombinationen, die gesucht werden müssen, sind fast unerschöpflich und können den Spieler stundenlang fesseln. Außer dem Effekt der Unterhaltung schult das Spiel sehr intensiv das logische Denken. Die Regeln sind sehr einfach und wurden in Heft 1 bereits ausführlich dargestellt. Der Rechner »denkt« sich eine Zahl aus, die vom Spieler zu finden ist. Im hier vorgestellten Beispiel wurde die Variante »4 aus 10 mit Dopplungen« gewählt, so daß sich das Zahlenspektrum von 0000 bis 9999 erstreckt. Durch den Spieler ist dem Rechner ein Angebot der vermuteten Zahl zu unterbreiten, welches bewertet wird. Es ergeben sich drei Reaktionsmöglichkeiten des Rechners:

a) keine Reaktion – die angebotenen Ziffern sind in der zu suchenden Zahl nicht enthalten,

b) ein Kreuz(Plus) – wird erteilt, wenn eine der angebotenen Ziffern in der gesuchten Zahl vorhanden ist, aber an falscher Stelle steht und

c) ein Stern – wenn eine der angebotenen Ziffern bereits an der richtigen Stelle steht.

Aus der Zahl der ausgegebenen Sterne und Kreuze geht aber nicht hervor, welche Ziffer mit welchem Symbol gemeint ist.
Durch gezielte Auswertung der Ein- und Ausgaben sind die nötigen Ziffern und Positionen zu finden. Die Zahl der möglichen Versuche wird vom Rechner begrenzt.
Aus diesen Regeln läßt sich bereits ein grober Ablaufplan entwerfen (Bild 1).
Der kritische Punkt an diesem Spiel ist die Bewertung. Mit ihr wird das Verhalten des Spielers beeinflußt, da an dieser Stelle das Angebot an Bewertungszeichen festgelegt wird. Das betrifft besonders den Fall mit doppelten Ziffern. Das Problem dabei ist die Bewertung nach hinten (von der Eingabe zum gesuchten Wert) beziehungsweise nach vorn (vom gesuchten Wert zur Eingabe).

Ein *Beispiel:*

| | a) | | | | b) | | | |
|---|---|---|---|---|---|---|---|---|
| gesucht | 1 | 2 | 3 | 2 | 1 | 2 | 3 | 2 |
| | ↑ | | | ↑ | ↓ | ↘ | | ↓ |
| eingegeben | 1 | 4 | 7 | 2 | 1 | 4 | 7 | 2 |
| Bewertung | * | | | * | * | + | | * |

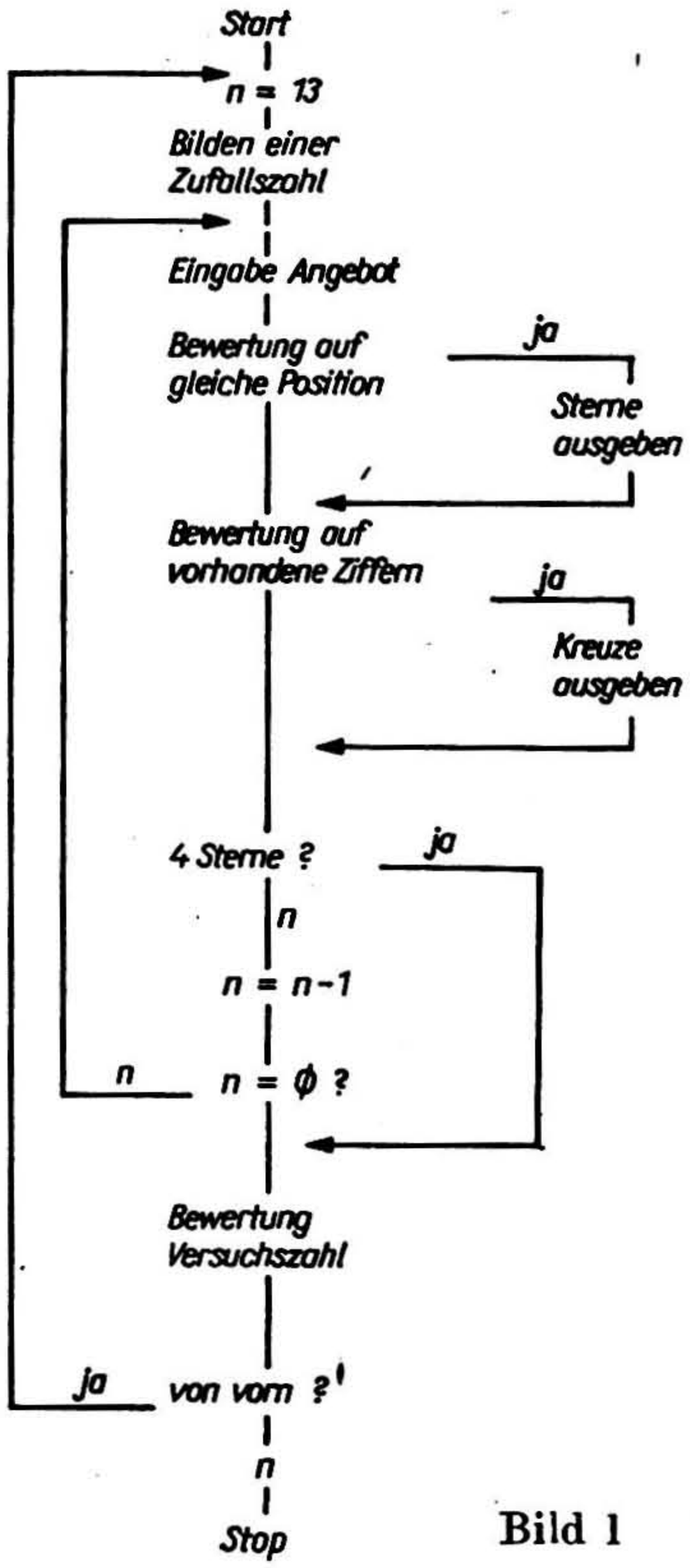



Bild 1

Da, um die Bewertung auf 4 Zeichen zu begrenzen, jede verglichene Stelle nur einmal eingesetzt wird, ergeben sich je nach Richtung des Vergleichs verschiedene Ergebnisse. Im Fall a) wird die 2 nur einmal gesucht und bewertet. Im Fall b) erfolgt die Suche zweimal mit zweimaliger Bewertung, wodurch der Eindruck erweckt wird, daß die 4 oder 7 noch mit richtig sind. Es ist allerdings auch eine Frage des gesamten Spielverlaufs, diese Bewertungen richtig zu deuten.

Das Programm wurde zur besseren Übersicht in Abschnitte unterteilt. Im Text wird immer die Nummer am Beginn des Abschnitts genannt.

Der Abschnitt Ø beginnt mit der Vorbereitung des Spiels. Es wird die Spielanleitung ausgegeben und durch Betätigen einer beliebigen Taste gestartet. Es erfolgt auch die Bildung der ersten gesuchten Zahl. Anschließend (Abschnitt 1) erfolgt die Initialisierung der eigentlichen Spielrunde. Da die Zufallszahl für die nächste Runde bereits während des Spiels gebildet wird, ist ein Umspeichern erforderlich.

In Abschnitt 2 erfolgt die Eingabe der angebotenen Zahl, wozu das HEX-Eingabeprogramm genutzt wird. Außerdem wird der »Sternzähler« rückgesetzt.

Der Abschnitt 3 entscheidet dann die Methode der Bewertung, wie sie zuvor beschrieben wurde. Er kann im Bedarfsfall weggelassen werden.

Im Abschnitt 4 wird der im Registerpaar BC stehende Wert aufgeteilt, um den Vergleich zu vereinfachen. Es steht dann je eine Ziffer in den Registern B bis E.

Die Abschnitte 5 bis 8 stellen den Vergleich dar. Sie unterliegen dem gleichen Aufbauprinzip mit folgenden Stufen:

- Bereitstellung Ziffer aus HL
- Vergleich auf gleiche Position
- Vergleich auf vorhanden.

War ein Vergleich erfolgreich, wird der Abschnitt verlassen, wodurch maximal 4 Bewertungszeichen entstehen können.

Im Abschnitt 9 erfolgt die Endeprüfung und anschließend die Bewertung bei Überziehen der Versuchszahl.

Im 11. Abschnitt kommt die Frage nach Rückkehr ins System oder Wiederholung des Spieles.

Der Abschnitt 12 realisiert die Bewertung der Versuchszahl. Es kann für jede Anzahl ein gesonderter Text ausgegeben werden. Das wird erreicht, indem mit der verbleibenden Versuchsanzahl eine Tabelle (Abschnitt 19) adressiert und ihr die Anfangsadresse des entsprechenden Textes entnommen wird.

```
0001                                    PN    MA
0002                                ;  MASTERMIND
0003                                ;
0004                                ;-----------------------------------------------000--
0005                                ;
0006                                       ORG   1000H
0007                                ;
0008                                ;  VORBEREITUNG SPIEL
0009    1000  3E0E                  ANF:   LD    A,0EH
0010    1002  CD4B11                       CALL  VIDEO                 ; LOESCHEN BILD
0011    1005  212612                       LD    HL,TXA
0012    1008  CD4511                       CALL  TEXT                  ; ANLEITUNG
0013    100B  CD5111                       CALL  TAST                  ; WARTEN AUF BEGINN
0014    100E  CD1A11                       CALL  ZUFA                  ; BILDEN START-ZUFALLSZAHL
0015                                ;
0016                                ;-----------------------------------------------001--
0017                                ;
0018                                ;  VORBEREITUNG 1 RUNDE
0019                                ;
0020    1011  3E0E                  MA:    LD    A,0EH
0021    1013  CD4B11                       CALL  VIDEO                 ; LOESCHEN BILD
0022    1016  3E0D                         LD    A,13
0023    1018  329213                       LD    (ZAEHL),A             ; ZAHL DER VERSUCHE
0024    101B  2A9013                       LD    HL,(BZUF)
0025    101E  228E13                       LD    (AZUF),HL             ; HOLEN AKT. ZAHL
0026                                ;
0027                                ;-----------------------------------------------002--
0028                                ;
0029                                ;  EINGABE DER ERMITTELTEN ZAHL
0030                                ;
0031    1021  CD4811                MC:    CALL  ADRIN                 ; 4 ZEICHEN EINGEBEN
0032    1024  E5                           PUSH  HL
0033    1025  CD3211                       CALL  ZUFB                  ; BILDEN NAECHSTE ZAHL
0034    1028  AF                           XOR   A
0035    1029  329313                       LD    (STERN),A             ; LOESCHEN STERN-ZAEHLER
0036    102C  3E09                         LD    A,9
0037    102E  CD4B11                       CALL  VIDEO                 ; TAB AUSGEBEN
0038                                ;
0039    1031  C1                           POP   BC                    ; EINGEG. WERT IN BC
0040    1032  2A8E13                       LD    HL,(AZUF)             ; GESUCHT. WERT IN HL
0041                                ;
0042                                ;-----------------------------------------------003--
0043                                ;
0044                                ;  VERTAUSCHEN VERGLEICHSWERTE (KANN EVT. ENTFALLEN)
0045                                ;
0046    1035  E5                           PUSH  HL
0047    1036  C5                           PUSH  BC
0048    1037  E1                           POP   HL
0049    1038  C1                           POP   BC
0050                                ;
0051                                ;   <BC> = GESUCHTER WERT
0052                                ;   <HL> = EINGEGEBENER WERT
0053                                ;
0054                                ;-----------------------------------------------004--
0055                                ;
0056                                ;  VERGLEICH DER WERTE
0057                                ;  1.   GLEICHHEIT
0058                                ;  2...4. VORHANDEN
0059                                ;
0060                                ;
0061                                ;  AUFTEILEN DES EINGEGEBENEN WERTES
0062
0063    1039  79                           LD    A,C
0064    103A  E60F                         AND   0FH
0065    103C  5F                           LD    E,A                   ; <E> EINER
0066    103D  79                           LD    A,C
0067    103E  CD1111                       CALL  ROT
0068    1041  57                           LD    D,A                   ; <D> ZEHNER
0069    1042  78                           LD    A,B
0070    1043  E60F                         AND   0FH
0071    1045  4F                           LD    C,A                   ; <C> HUNDERTER
0072    1046  78                           LD    A,B
0073    1047  CD1111                       CALL  ROT
0074    104A  47                           LD    B,A                   ; <B> TAUSENDER
0075                                ;
0076                                ;.....................................................005....
```

```
0077                            ;
0078                            ;  VERGLEICH  EINERSTELLE
0079                            ;
0080    104B  7D                        LD    A,L
0081    104C  E60F                      AND   0FH
0082    104E  BB                        CMP   E
0083    104F  CCFA10                    CAZ   UPS                       ;GLEICH
0084    1052  2810                      JRZ   VZ-#
0085    1054  BA                        CMP   D
0086    1055  CC0911                    CAZ   UPK
0087    1058  280A                      JRZ   VZ-#
0088    105A  B9                        CMP   C
0089    105B  CC0911                    CAZ   UPK
0090    105E  2804                      JRZ   VZ-#
0091    1060  B8                        CMP   B
0092    1061  CC0911                    CAZ   UPK
0093                            ;
0094                            ;.....................................................006..
0095                            ;
0096                            ;  VERGLEICH  ZEHNERSTELLE
0097                            ;
0098    1064  7D                VZ:     LD    A,L
0099    1065  CD1111                    CALL  ROT
0100    1068  BA                        CMP   D
0101    1069  CCFA10                    CAZ   UPS
0102    106C  2810                      JRZ   VH-#
0103    106E  BB                        CMP   E
0104    106F  CC0911                    CAZ   UPK
0105    1072  280A                      JRZ   VH-#
0106    1074  B9                        CMP   C
0107    1075  CC0911                    CAZ   UPK
0108    1078  2804                      JRZ   VH-#
0109    107A  B8                        CMP   B
0110    107B  CC0911                    CAZ   UPK
0111                            ;
0112                            ;.....................................................007...
0113                            ;
0114                            ;  VERGLEICH  HUNDERTERSTELLE
0115                            ;
0116    107E  7C                VH:     LD    A,H
0117    107F  E60F                      AND   0FH
0118    1081  B9                        CMP   C
0119    1082  CCFA10                    CAZ   UPS
0120    1085  2810                      JRZ   VT-#
0121    1087  BB                        CMP   E
0122    1088  CC0911                    CAZ   UPK
0123    108B  280A                      JRZ   VT-#
0124    108D  BA                        CMP   D
0125    108E  CC0911                    CAZ   UPK
0126    1091  2804                      JRZ   VT-#
0127    1093  B8                        CMP   B
0128    1094  CC0911                    CAZ   UPK
0129                            ;
0130                            ;.....................................................008..
0131                            ;
0132                            ;  VERGLEICH  TAUSENDERSTELLE
0133                            ;
0134    1097  7C                VT:     LD    A,H
0135    1098  CD1111                    CALL  ROT
0136    109B  B8                        CMP   B
0137    109C  CCFA10                    CAZ   UPS
0138    109F  2810                      JRZ   VX-#
0139    10A1  B9                        CMP   C
0140    10A2  CC0911                    CAZ   UPK
0141    10A5  280A                      JRZ   VX-#
0142    10A7  BA                        CMP   D
0143    10A8  CC0911                    CAZ   UPK
0144    10AB  2804                      JRZ   VX-#
0145    10AD  BB                        CMP   E
0146    10AE  CC0911                    CAZ   UPK
0147                            ;
0148                            ;-----------------------------------------------------009---
0149                            ;
0150                            ;  AUSWERTUNG ZAEHLERSTAENDE
0151                            ;
0152    10B1  3E1E              VX:     LD    A,1EH
```

```
0153    10B3  CD4B11          CALL  VIDEO                   ; NEUE ZEILE
0154    10B6  3A9313          LD    A,(STERN)
0155    10B9  FE04            CMP   4
0156    10BB  CAE510          JPZ   ENDE                    ; ALLE RICHTIG
0157                  ;
0158    10BE  3A9213          LD    A,(ZAEHL)
0159    10C1  3D              DEC   A                       ; VERSUCH - 1
0160    10C2  329213          LD    (ZAEHL),A
0161    10C5  C22110          JPNZ  MC                      ; NAECHSTER VERSUCH
0162                  ;
0163                  ;-----------------------------------------------------010---
0164                  ;
0165                  ;  VERSUCHSZAHL UEBERSCHRITTEN
0166                  ;
0167    10C8  214B13          LD    HL,TXB
0168    10CB  CD4511          CALL  TEXT
0169    10CE  2A8E13          LD    HL,(AZUF)
0170    10D1  CD4E11          CALL  VIDAD                   ; GESUCHTER WERT
0171                  ;
0172                  ;-----------------------------------------------------011---
0173                  ;
0174                  ;  ENDE ODER FORTSETZUNG ?
0175                  ;
0176    10D4  217513  STOP:   LD    HL,TXC
0177    10D7  CD4511          CALL  TEXT
0178    10DA  CD5111          CALL  TAST
0179    10DD  FE4A            CMP   'J'                     ; JA ?
0180    10DF  CA1110          JPZ   MA                      ; VON VORN
0181    10E2  C33800          JMP   38H                     ; ENDE
0182                  ;
0183                  ;-----------------------------------------------------012---
0184                  ;
0185                  ;  RICHTIGEN WERT GEFUNDEN  ( <ZAEHL>= 1...13 )
0186                  ;
0187    10E5  3A9213  ENDE:   LD    A,(ZAEHL)
0188    10E8  3D              DEC   A                       ; 0...12
0189    10E9  87              ADD   A                       ; MAL 2
0190    10EA  215411          LD    HL,TTAB                 ; TEXT-TABELLE
0191    10ED  0600            LD    B,0
0192    10EF  4F              LD    C,A
0193    10F0  09              ADD   HL,BC
0194    10F1  5E              LD    E,M
0195    10F2  23              INC   HL
0196    10F3  56              LD    D,M                     ; TEXT-ADRESSE IN DE
0197    10F4  EB              EX    DE,HL
0198    10F5  CD4511          CALL  TEXT
0199    10F8  18DA            JR    STOP-#
0200                  ;
0201                  ;-----------------------------------------------------013---
0202                  ;
0203                  ;  UP    AUSGABE STERN
0204                  ;
0205    10FA  F5      UPS:    PUSH  AF
0206    10FB  3E2A            LD    A,'*'
0207    10FD  CD4B11          CALL  VIDEO
0208    1100  3A9313          LD    A,(STERN)
0209    1103  3C              INC   A
0210    1104  329313          LD    (STERN),A
0211    1107  F1              POP   AF
0212    1108  C9              RET
0213                  ;
0214                  ;-----------------------------------------------------014---
0215                  ;
0216                  ;  UP    AUSGABE KREUZ
0217                  ;
0218    1109  F5      UPK:    PUSH  AF
0219    110A  3E2B            LD    A,'+'
0220    110C  CD4B11          CALL  VIDEO
0221    110F  F1              POP   AF
0222    1110  C9              RET
0223                  ;
0224                  ;-----------------------------------------------------015---
0225                  ;
0226                  ;  UP    ROTATION RECHTS
0227                  ;
0228    1111  CB3F    ROT:    SRL   A
```

```
0229    1113  CB3F              SRL   A
0230    1115  CB3F              SRL   A
0231    1117  CB3F              SRL   A
0232    1119  C9                RET
0233                      ;
0234                      ;-------------------------------------------------016---
0235                      ;
0236                      ;  BILDEN DER START-ZUFALLSZAHL
0237                      ;
0238    111A  0604        ZUFA: LD    B,4
0239    111C  219013            LD    HL,BZUF
0240    111F  3A9413            LD    A,(WERT)
0241    1122  4F                LD    C,A                    ; ZAHLENBEREICH
0242                      ;
0243    1123  ED5F        ZUFAB: LD   A,R
0244    1125  91          ZUFAC: SUB  C
0245    1126  30FD,             JRNC  ZUFAC-#
0246    1128  81                ADD   C
0247    1129  ED6F              RLD
0248    112B  23                INC   HL
0249    112C  ED6F              RLD
0250    112E  2B                DEC   HL
0251    112F  10F2              DJNZ  ZUFAB-#
0252    1131  C9                RET
0253                      ;
0254                      ;-------------------------------------------------017---
0255                      ;
0256                      ;  ZUFALLSZAHL B
0257                      ;
0258    1132  219013      ZUFB: LD    HL,BZUF
0259    1135  3A9413            LD    A,(WERT)
0260    1138  4F                LD    C,A
0261    1139  ED5F              LD    A,R
0262    113B  91          ZUFBA: SUB  C
0263    113C  30FD              JRNC  ZUFBA-#
0264    113E  81                ADD   C
0265    113F  ED6F              RLD
0266    1141  23                INC   HL
0267    1142  ED6F              RLD
0268    1144  C9                RET
0269                      ;
0270                      ;-------------------------------------------------018---
0271                      ;
0272                      ;  VERBINDUNG ZUM BETRIEBSSYSTEM
0273                      ;
0274    1145  C30000      TEXT: JMP   000                    ; TEXTAUSGABE
0275    1148  C30000      ADRIN: JMP  000                    ; EINGABE 4 ZEICHEN HEX
0276    114B  C30000      VIDEO: JMP  000                    ; AUSG. 1 ZEICHEN AUF BS
0277    114E  C30000      VIDAD: JMP  000                    ; AUSG. 4 ZEICHEN HEX
0278    1151  C30000      TAST: JMP   000                    ; EING. 1 ZEICHEN
0279                      ;
0280                      ;-------------------------------------------------019---
0281                      ;
0282                      ;  TEXT-TABELLE FUER SPIELBEWERTUNG
0283                      ;
0284    1154  6E11        TTAB: DA    T1                     ; SCHLECHTESTE LEISTUNG
0285    1156  8211              DA    T2
0286    1158  8211              DA    T2
0287    115A  9B11              DA    T3
0288    115C  B211              DA    T4
0289    115E  B211              DA    T4
0290    1160  C011              DA    T5
0291    1162  CC11              DA    T6
0292    1164  E111              DA    T7
0293    1166  FA11              DA    T8
0294    1168  FA11              DA    T8
0295    116A  0D12              DA    T9
0296    116C  0D12              DA    T9                     ; BESTE LEISTUNG
0297                      ;
0298                      ;.................................................020...
0299                      ;
0300                      ;  TEXTE ZUR BEWERTUNG
0301                      ;
0302    116E  41554553    T1:   DB    'AUESSERST SCHWACH !',3
        1172  53455253
        1176  54205343
```

```
                117A  48574143
                117E  48202103
0303            1182  4E4F4348  T2:     DB    'NOCH ANNEHMBARE LEISTUNG',3
                1186  20414E4E
                118A  45484D42
                118E  41524520
                1192  4C454953
                1196  54554E47
                119A  03
0304            119B  5343484F  T3:     DB    'SCHON BESSER GEWESEN !',3
                119F  4E204245
                11A3  53534552
                11A7  20474557
                11AB  4553454E
                11AF  202103
0305            11B2  4D454852  T4:     DB    'MEHR UEBEN !!',3
                11B6  20554542
                11BA  454E2021
                11BE  2103
0306            11C0  47555420  T5:     DB    'GUT GEMACHT',3
                11C4  47454D41
                11C8  43485403
0307            11CC  53454852  T6:     DB    'SEHR GUTE LEISTUNG !',3
                11D0  20475554
                11D4  45204C45
                11D8  49535455
                11DC  4E472021
                11E0  03
0308            11E1  42495454  T7:     DB    'BITTE ALLEIN SPIELEN !!!',3
                11E5  4520414C
                11E9  4C45494E
                11ED  20535049
                11F1  454C454E
                11F5  20212121
                11F9  03
0309            11FA  44415320  T8:     DB    'DAS WAR ZUFALL !!!',3
                11FE  57415220
                1202  5A554641
                1206  4C4C2021
                120A  212103
0310            120D  554E4D4F  T9:     DB    'UNMOEGLICH ERREICHBAR !!',3
                1211  45474C49
                1215  43482045
                1219  52524549
                121D  43484241
                1221  52202121
                1225  03
0311                          ;
0312                          ;-----------------------------------------------------021---
0313                          ;
0314                          ;  ZENTRALE TEXTE
0315                          ;
0316            1226  45532049  TXA:    DB    'ES IST EINE 4-STELLIGE ZAHL',36Q
                122A  53542045
                122E  494E4520
                1232  342D5354
                1236  454C4C49
                123A  4745205A
                123E  41484C1E
0317            1242  5A552046        DB    'ZU FINDEN. DAFUER STEHEN 13 ',36Q
                1246  494E4445
                124A  4E2E2044
                124E  41465545
                1252  52205354
                1256  4548454E
                125A  20313320
                125E  1E
0318            125F  56455253        DB    'VERSUCHE ZUR VERFUEGUNG.',36Q
                1263  55434845
                1267  205A5552
                126B  20564552
                126F  46554547
                1273  554E472E
                1277  1E
0319            1278  53544548        DB    'STEHT IN DER EINGEGEBENEN ZAHL',36Q
                127C  5420494E
```

```
                1280  20444552
                1284  2045494E
                1288  47454745
                128C  42454E45
                1290  4E205A41
                1294  484C1E
0320            1297  42455245       DB    'BEREITS EINE ZIFFER AN RICH-',36Q
                129B  49545320
                129F  45494E45
                12A3  205A4946
                12A7  46455220
                12AB  414E2052
                12AF  4943482D
                12B3  1E
0321            12B4  54494745       DB    'TIGER STELLE, SO ERSCHEINT',36Q
                12B8  52205354
                12BC  454C4C45
                12C0  2C20534F
                12C4  20455253
                12C8  43484549
                12CC  4E541E
0322            12CF  45494E20       DB    'EIN STERN. IST DIE ZIFFER NUR',36Q
                12D3  53544552
                12D7  4E2E2049
                12DB  53542044
                12DF  4945205A
                12E3  49464645
                12E7  52204E55
                12EB  521E
0323            12ED  564F5248       DB    'VORHANDEN, ERSCHEINT EIN KREUZ.',36Q
                12F1  414E4445
                12F5  4E2C2045
                12F9  52534348
                12FD  45494E54
                1301  2045494E
                1305  204B5245
                1309  555A2E1E
0324            130D  2A20554E       DB    '* UND + GEBEN NICHT DIE PO-',36Q
                1311  44202B20
                1315  47454245
                1319  4E204E49
                131D  43485420
                1321  44494520
                1325  504F2D1E
0325            1329  53495449           DB    'SITION DER BEWERTETEN ZIFFER AN!',36Q
                132D  4F4E2044
                1331  45522042
                1335  45574552
                1339  54455445
                133D  4E205A49
                1341  46464552
                1345  20414E21
                1349  1E
0326            134A  03                 DB    3
0327            134B  45494E20   TXB:    DB    'EIN HOFFNUNGSLOSER FALL !!',36Q
                134F  484F4646
                1353  4E554E47
                1357  534C4F53
                135B  45522046
                135F  414C4C20
                1363  21211E
0328            1366  52494348           DB    'RICHTIG WAR : ',3
                136A  54494720
                136E  57415220
                1372  3A2003
0329            1375  1E         TXC:    DB    36Q
0330            1376  564F4E20           DB    'VON VORN ? (JA/NEIN) : ',3
                137A  564F524E
                137E  203F2028
                1382  4A412F4E
                1386  45494E29
                138A  203A2003
0331                              ;
0332                              ;-----------------------------------------------022--
0333                              ;
0334                              ;   R A M
```

```
0335
0336    138E              AZUF:   BER   2                    ; AKTUELLE ZAHL
0337    1390              BZUF:   BER   2                    ; NAECHSTE ZAHL
0338    1392              ZAEHL:  BER   1                    ; ZAHL DER VERSUCHE
0339    1393              STERN:  BER   1                    ; ZAEHLER STERNE
0340                      ;
0341    1394  0A          WERT:   DB    10  ; ZAHLENBEREICH
0342                      ;
0343                      ;-----------------------------------------------------023--
0344                      ;
0345                      ;
0346                      ;
0347                      ;
0348                      ;
0349                      ;
0350                            END

KEINE SYNTAX-FEHLER CRAS 4200-K1520
```

Die Zufallszahlen werden in den Abschnitten 16 und 17 gebildet. Dabei ist die Startzufallszahl mit Vorsicht zu betrachten. Als Basis dient das Refresh-Register des Prozessors, das nach jedem Befehl um eins erhöht wird. Es bringt einen Wert zwischen 0 und 127. Von diesem Wert wird so lange die Konstante »WERT« subtrahiert, bis das Ergebnis kleiner als diese ist.
Auf diese Weise kann der Bereich der Ziffern festgelegt werden. Nach viermaliger Operation stehen die 4 Stellen der Zufallszahl zur Verfügung und werden in BZUF abgelegt. Der Nachteil der Startzufallszahl besteht darin, daß zwischen den Entnahmen aus dem R-Register keine längeren Befehlsfolgen liegen, so daß zwar die erste Stelle zufällig ist, die weiteren sich aber in ihrer Nähe befinden.
Günstiger ist das bei der Zufallszahl B. Dieses Programm bildet nur eine Stelle, welche auf den Speicherplatz BZUF gebracht wird. Dazu werden zuvor die 16 Bit um 4 Bit nach links verschoben und die neue Stelle unten angesetzt. Dieses Durchschieben erfolgt nach jeder Eingabe. Da der Rechner während der Tastaturbedienung mehrere tausend Befehle abarbeitet (durch die Zeitschleifen der Entprellung), kann man den damit erzeugten Wert als zufällig betrachten.
Der Abschnitt 18 enthält die Verbindungen zu den Programmen des Betriebssystems. Von dort ist mit »RET« zurückzukehren.
Das Programm ist auf dem in Heft 5 dieser Reihe vorgestellten Betriebssystem lauffähig und wurde darauf getestet.

## Literatur


[1] Gutzer, H.: Das MASTER-MIND-SPIEL – der K 1003 als Herausforderer. – In: Kleinstrechner-TIPS, Heft 1. – Leipzig: Fachbuchverlag, 1984. – S. 44



Autor:

*Dr.-Ing. Gert Schönfelder*
Problemanalytiker im Rechenzentrum der Sektion Informationsverarbeitung der Ingenieurhochschule Dresden

# Der Computer als »Hellseher«

Sicher kennen Sie das schöne Spielchen, bei dem man von einem klugen Menschen aufgefordert wird, sich eine Zahl zu denken und mit dieser Zahl eine Reihe von Rechenoperationen auszuführen. Er läßt sich dann nur das Ergebnis der Rechnungen nennen und teilt dann nach sehr kurzer Zeit zum Erstaunen der nicht eingeweihten Anwesenden mit, welche Zahl man sich gemerkt hatte.

Ein Kleincomputer kann das natürlich auch. Wir müssen ihm nur in Form eines Programms vorher sagen, wie er dabei vorzugehen hat. Dies ist – wie wir sehen werden – gar nicht so schwierig.

Wir wollen nun ein solches Spielprogramm aufstellen und dabei die Gelegenheit nutzen, einige Grundsätze kennenzulernen, die man bei der Erarbeitung derartiger Computerspiele stets beachten sollte. Außerdem wollen wir das Programm in der Programmiersprache PASCAL formulieren, um daran einige wichtige Eigenschaften dieser modernen Computersprache kennenzulernen.

(Die Sprache PASCAL kann bei den Bürocomputern A 5110 bis A 5130 bereits genutzt werden. Für den Kleincomputer KC 85/2 ist vorgesehen, einen Zusatzbaustein PASCAL zur Verfügung zu stellen.)

Wenn man ein Zahlenratespiel programmieren will, muß man sich zunächst einmal darüber klar werden, wie es der Spielführer macht, wenn er die von uns gedachte Zahl so schnell »errät«, (In Wirklichkeit berechnet er sie.)

Betrachten wir dazu ein Beispiel: Wir werden aufgefordert, eine Zahl zu merken, sie mit 3 zu multiplizieren, dann 5 zu addieren, davon 8 zu subtrahieren und schließlich von dem Ganzen dann noch die Hälfte zu nehmen. Das Resultat dieser Rechnung soll dann genannt werden.

Wenn wir die gedachte Zahl ermitteln wollen, brauchen wir sie nur mit $x$ und das genannte Resultat mit $y$ zu bezeichnen. Vollziehen wir die oben angegebenen Aufforderungen mit diesen beiden Variablen Schritt für Schritt nach, so erhalten wir die folgende Gleichung:

$$((3 \cdot x + 5) - 8) : 2 = y,$$

oder vereinfacht

$$\frac{3x - 3}{2} = y,$$

woraus die unbekannte Größe $x$ leicht berechnet werden kann:

$$x = \frac{2}{3} \cdot y + 1.$$

Damit ist der erste wichtige Schritt zur Aufstellung unseres Spielprogrammes abgeschlossen. Wir wissen, wie wir aus der vom Mitspieler genannten Zahl zu der von ihm ursprünglich gemerkten

Zahl kommen: Das Ergebnis braucht nur mit 2/3 multipliziert zu werden und dazu ist dann noch eine 1 hinzuzufügen.
Nun muß diese Erkenntnis für den Computer aufbereitet werden, damit er mit einem Menschen spielen kann. Wir müssen also dafür sorgen, daß der Computer alle vom Spielführer ausgeführten Aktivitäten übernimmt und sie in gleicher Weise durchführt wie er. Dabei müssen wir natürlich all die Eigenschaften berücksichtigen, in denen sich ein Computer vom Menschen unterscheidet. Was muß der Computer also alles tun?
Zunächst einmal muß der Computer dem Mitspieler die zu lösende Aufgabe mitteilen. Als Mensch würde man die Aufgabe in Worten formulieren und sie schön langsam und verständlich sprechen, damit sie der Mitspieler auch verstehen kann. Solange uns keine sprechenden Kleincomputer zur Verfügung stehen, muß der Computer die Aufgabe schriftlich formulieren und sie als Text auf dem Display erscheinen lassen. Auch hier sollte man berücksichtigen, daß der Mensch normalerweise kein Schnelleser und Schnelldenker ist, so daß man die Zeilen auf dem Bildschirm nicht zu lang gestaltet und sie auch einige Zeit stehen läßt, damit der Spieler den Sinn des Textes erfassen und die Tätigkeiten ausführen kann, zu denen er aufgefordert wird.
Ist die Aufgabe dem Mitspieler mitgeteilt, so muß die Aufforderung zur Bekanntgabe des erreichten Ergebnisses folgen. Dieses Ergebnis muß über die Tastatur in den Computer eingegeben werden. Nachdem dies geschehen ist, kann der Computer nach der vorn angegebenen Formel die Ausgangszahl ermitteln und sie schließlich zum Erstaunen der Mitspieler über das Display ausgeben.
Damit haben wir den folgenden Grobablauf unseres Spielprogrammes (s. Ablaufplan 1).
Nun könnte es ja vorkommen, daß der Mitspieler mit diesem einen Spiel nicht zufrieden ist und das Spielchen wiederholen will. (Vielleicht hofft er insgeheim, den Computer doch irgendwie überlisten zu können.) Also werden wir in unserem Programm die Möglichkeit vorsehen, das Spiel zu wiederholen.
Das kann dadurch geschehen, daß wir in unser Programm eine Zusatzfrage einbauen, aus deren Antwort hervorgehen muß, ob eine Wiederholung des Spiels gewünscht wird oder nicht. Außerdem wollen wir unser Spiel noch dadurch etwas attraktiver gestalten, daß wir den Mitspieler zu Beginn des

*Ablaufplan 1*

```
AUSGABE: AUFGABENTEXT
EINGABE: ERGEBNIS DER RECHNUNG
BERECHNUNG DER GEDACHTEN ZAHL
AUSGABE: GEDACHTE ZAHL
```

*Ablaufplan 2*

```
AUSGABE: FREUNDLICHE BEGRUESZUNG
SOLANGE EIN SPIEL GEWUENSCHT WIRD
            AUSGABE: AUFGABENTEXT
            EINGABE: ERGEBNIS DER RECHNUNG
            BERECHNUNG DER GEDACHTEN ZAHL
            AUSGABE: GEDACHTE ZAHL
AUSGABE:  VERABSCHIEDUNG DES MITSPIELERS
```

Spieles durch den Computer freundlich begrüßen lassen. (Dies trägt dazu bei, daß der Mensch seine Scheu vor dem technischen Wunderwerk Computer verliert, und es werden dann vielleicht auch einmal Oma und Opa ein Spielchen wagen, die sonst vielleicht gesagt hätten: »Ach laßt mich doch zufrieden mit diesem modernen Zeug!«) Man sollte überhaupt lieber einmal mehr als einmal zuwenig ein paar nette und aufmunternde Worte während eines Spiels am Bildschirm erscheinen lassen. Der Computer erhält dadurch menschlichere Züge, und das Spiel oder die Arbeit mit ihm bereitet einem gleich viel mehr Freude!

Mit diesen Ergänzungen haben wir nunmehr den folgenden Ablauf für unser einfaches Spielprogramm erhalten (s. Ablaufplan 2).

Das zugehörige Programm wird auf Seite 53 begonnen. Es wurde, wie bereits eingangs erwähnt, in der Programmiersprache PASCAL formuliert, um an diesem Beispiel einige wesentliche Unterschiede zwischen den Sprachen PASCAL und BASIC dokumentieren zu können. Für diejenigen, die die Sprache BASIC beherrschen, wird es nicht schwer sein, das vorgestellte Programm in BASIC umzuschreiben. Man sollte dies ruhig tun, den es stellt eine gute Programmierübung für BASIC dar und fördert gleichzeitig das Verständnis für die anspruchsvollere, aber auch leistungsfähigere Programmiersprache PASCAL.

Die einzelnen Schritte des Programmes sollen anschließend etwas ausführlicher kommentiert werden.

Jedes PASCAL-Programm beginnt mit der Eröffnungsformel

PROGRAM name (liste von dateien);

wobei für name irgendein das Programm charakterisierender Name eingesetzt werden darf und wobei in der liste von dateien alle diejenigen Dateien aufgeführt werden müssen, die für die Abarbeitung des Programmes erforderlich sind. In unserem Falle haben wir das Programm PASGAME 1 genannt (PAS, weil es in PASCAL formuliert wurde, GAME (engl.) game, das Spiel), weil es sich um ein Spielprogramm handelt. Es können natürlich auch beliebige andere und nettere Namen erfunden werden. – Die für die Abarbeitung des Programmes erforderlichen Dateien sind die Eingabedatei INPUT und die Ausgabedatei OUTPUT. Würden noch weitere Dateien benötigt, so müßten sie ebenfalls innerhalb der runden Klammern, durch Kommata voneinander getrennt, aufgeführt werden.

Die Programmzeilen 200 und 300 enthalten einen sogenannten Kommentar. Kommentare sind Mitteilungen an den Nachnutzer eines Programmes, um das Programm zu erläutern, um besondere Hinweise für den Nutzer zu geben u.a.m. Sie beginnen mit dem Zeichenpaar (* und enden mit dem Zeichenpaar *). Alles, was zwischen diesen beiden Zeichenpaaren steht, wird vom Computer nicht zur Kenntnis genommen. In den Zeilen 400 und 500 werden die im Programm auftretenden Variablen vereinbart. Im Gegensatz zur Sprache BASIC müssen in einem PASCAL-Programm *alle* im Programm verwendeten Größen vereinbart werden. Damit wird der Computer darüber informiert, wie viele Größen in dem Programm auftreten, von welcher Art sie sind, und er kann von vornherein den entsprechenden Speicherraum für diese Größen reservieren. Tritt dann im Programm eine Größe auf, die nicht vereinbart wurde, oder wird einer Größe während der Programmabarbeitung ein Wert zugewiesen, der nicht dem vereinbarten Typ entspricht, so gibt der Rechner eine entsprechende

Fehlermeldung aus und stoppt die weitere Abarbeitung des Programmes. Dieser Zwang zur sinnvollen Vereinbarung aller auftretenden Größen ist für eine systematische Programmentwicklung von außerordentlich großer Bedeutung. In unserem Falle werden in Programmzeile 400 die beiden Variablen $X$ und $Y$ als reelle Zahlen festgelegt, während für die Variable $Z$ ein beliebiges alphanumerisches Zeichen eingesetzt werden darf.
Neben den Variablen-Vereinbarungen, die durch das Wörtchen VAR eingeleitet werden, gibt es in PASCAL noch eine ganze Reihe weiterer Vereinbarungsmöglichkeiten, die in einer festgelegten Aufeinanderfolge programmiert werden müssen. Darüber soll in einem späteren Artikel einiges ausgeführt werden. Die Programmzeilen 600 bis 1400 enthalten ein sogenanntes Unterprogramm (eine Prozedur), das im Verlaufe unseres Spielprogramms mehrfach benötigt wird. Es veranlaßt den Rechner, wie dies in dem Kommentar (Zeilen 700 bis 1000) erläutert wird, an bestimmten Programmstellen einige Zeit zu warten, damit die ausgegebenen Bildschirmtexte gelesen und verarbeitet werden können.
In diesem Unterprogramm wird mit Hilfe einer Laufanweisung eine für die sonstige Programmabarbeitung sinnlose Anweisung $K := K$; fünfzigmal wiederholt. Soll die Wartezeit des Display-Bildes länger dauern, so müßte man nur die Formulierung
FOR K := 1 TO 50 ersetzen durch FOR K := 1 TO 100. Dies würde einer Verdoppelung der bisherigen Wartezeit entsprechen. Die für die Prozedur WARTEN erforderliche Variable $K$ wurde in der Zeile 1100 vereinbart.
Durch das Wörtchen BEGIN in Zeile 1500 wird nun das eigentliche Hauptprogramm eröffnet. Die beiden Anweisungen WRITELN; in Zeile 1600 veranlassen, daß der auszugebende Text auf dem Bildschirm nicht gleich in der obersten Zeile beginnt, sondern daß erst einmal zwei Zeilen freigelassen werden. Derartige Leerzeilen finden sich dann auch in den Programmzeilen 1900, 2300, 2900, 3200 usw.
Damit der ausgegebene Text auch verstanden werden kann, sind in den Programmzeilen 1900, 2300 usw. durch WARTEN kleine Aufenthalte im Programmablauf vorgesehen.
Die Programmzeilen 1600 bis 2300 veranlassen den Computer, den angegebenen Begrüßungstext über den Bildschirm auszugeben. Dabei ist wegen des Verschlüsselungswortes WRITELN jeder zwischen (' und '); untergebrachte Text dann auf einer Zeile des Bildschirms sichtbar.
Im Befehl 2400 erwartet der Rechner die Eingabe irgendeines alphanumerischen Zeichens Z (Z wurde im Vereinbarungsteil als CHAR deklariert). Da der Mitspieler seine Antwort vielleicht nicht gleich mit dem ersten Zeichen auf der Eingabetastatur eintippen, sondern vielleicht erst einige Leerzeichen tippen wird, wurde der Eingabebefehl in der Form

```
REPEAT READ (Z) UNTIL Z <> ' ';
```

formuliert. Diese Befehlsfolge besagt, daß der Rechner die Eingabe von je einem Zeichen so lange zu wiederholen hat (REPEAT), bis ein von einem Leerzeichen verschiedenes Zeichen erscheint (UNTIL Z <> ' '). Dieses erste von einem Leerzeichen verschiedene Zeichen speichert der Computer dann in dem für Z vorgesehenen Speicherbereich.
Mit

```
WHILE Z = 'J' DO
```

(Programmzeile 2500) wird nun veranlaßt, daß die mit BEGIN (Zeile 2600) bis END (Zeile 6200) zusammengefaßte

Befehlsfolge so lange abgearbeitet wird, bis einmal für Z ein von J verschiedenes Zeichen eingegeben wird.
Es ist leicht zu erkennen, daß durch die Programmzeilen 2700 bis 4300 die Aufgabenformulierung am Bildschirm erscheinen wird. Man könnte diese Aufgabenstellung natürlich wesentlich kürzer und sachlicher formulieren und damit sowohl Speicherplatz als auch Rechenzeit einsparen. Aber gerade die auflockernden Bemerkungen und Fragen sind es ja, die das Spielen mit einem Computer angenehm machen sollen.
Nach dem Befehl 4400 erwartet der Rechner die Eingabe des vom Mitspieler erzielten Ergebnisses über die Eingabetastatur. Er berechnet dann (Programmzeile 4500) die Ausgangszahl und gibt das Resultat seiner Berechnungen mit einigen freundlichen Bemerkungen (Programmzeilen 4600 bis 5500) aus. Während in den bisherigen Ausgabebefehlen immer nur Texte auf dem Bildschirm erscheinen sollten, soll in Programmzeile 5100 neben dem Text ES WAR DIE ZAHL auch der Zahlenwert von *X* gezeigt werden.
Dabei gilt für die Ausgabe folgende Regelung: Soll durch einen WRITE- oder WRITELN-Befehl eine beliebige Folge von Zeichen ausgegeben werden, so ist diese Zeichenfolge durch Apostrophe einzuschließen. (Daher: Writeln (' ES WAR DIE ZAHL ' ...)) Soll hingegen der für eine Variable gespeicherte Wert ausgegeben werden, so darf nur diese Variable im WRITE- oder WRITELN-Befehl angegeben werden. Dabei sind mehrere unterschiedliche Ausgabeforderungen durch Kommata voneinander zu trennen. (Aus diesem Grunde wird der obige Ausgabebefehl wie folgt fortgesetzt:

```
WRITELN
  (' ES WAR DIE ZAHL', X)
```

Durch die noch folgende Zusatzangabe :3:0 wird der Rechner veranlaßt, den Zahlenwert von *X* rechtsbündig innerhalb von 3 Druckstellen auszugeben. Die folgende Null bedeutet, daß keine Kommastellen ausgegeben werden sollen (die gedachte Zahl sollte ja eine ganze Zahl sein).
Schließlich sei noch eine Bemerkung zum Unterschied zwischen der Aufforderung WRITE und der Aufforderung WRITELN gemacht. Wird eine Ausgabe nur durch das Wort WRITE (...) angewiesen, so verbleibt die Ausgabe-Steuerung auch nach dem Erscheinen des Inhalts des WRITE-Befehls auf dem Bildschirm noch in derselben Zeile, und die folgenden Ausgabetexte schließen sich unmittelbar an den letzten Text an. Bei WRITELN hingegen wird *nach* der Ausgabe des Textes oder der Variablen sofort zu einer neuen Zeile übergegangen, so daß die nachfolgende Ausgabeanweisung auf einer neuen Zeile erscheint.
Auf Grund der Programmzeilen 5700 bis 5900 erkundigt sich nun der Computer, ob das Spiel wiederholt werden soll, und erwartet (Programmzeile 6100) die Eingabe der entsprechenden Antwort. Lautet diese *J*, so wird die gesamte Befehlsfolge von Befehl 2700 bis Anweisung 6100 abgearbeitet, wobei natürlich die vom Mitspieler eingegebenen Werte ganz anders sein dürfen als beim vorangegangenen Spiel. Wird jedoch ein *N* oder ein anderes von *J* verschiedenes Zeichen als Antwort auf die letzte Frage eingegeben, so deutet dies der Rechner als Aufforderung, das Spiel nunmehr abzubrechen. Er geht daher zu den folgenden Programmschritten 6300 ff. über und verabschiedet sich mit freundlichen Worten von seinem Mitspieler.

```
PROGRAM PASGAME 1 (INPUT, OUTPUT);                                          100

(* DIALOGPROGRAMM FUER EIN EINFACHES ZAHLENRATESPIEL ZWISCHEN               200
EINEM COMPUTER UND EINEM MENSCHEN .                                  *)     300
VAR      X, Y            : REAL ;                                           400
         Z               : CHAR ;                                           500

PROCEDURE WARTEN ;                                                          600
  (* DIE PROZEDUR WARTEN DIENT ZUM FESTHALTEN EINES BILD-                   700
  SCHIRMTEXTES FUER EINE BESTIMMTE ZEIT, DAMIT DER MIT-                     800
  SPIELER DIE INFORMATIONEN AUFNEHMEN UND VERARBEITEN                       900
  KANN .                                                             *)    1000
  VAR      K               : INTEGER ;                                      1100
  BEGIN                                                                     1200
  FOR K :=                 1 TO 5Ø DO K := K ;                              1300
  END ;          (*        ENDE DER PROZEDUR WARTEN .                 *)    1400

BEGIN            (*        ANFANG DES SPIELPROGRAMMS .                *)    1500
  WRITELN ;                WRITELN ;                                        1600
  WRITELN        ('        ICH BEGRUESZE SIE SEHR HERZLICH ') ;             1700
  WRITELN        ('        AN MEINEM BILDSCHIRM . ') ;                      1800
  WARTEN ;                 WRITELN ;                                        1900
  WRITELN        ('        WOLLEN SIE MIT MIR EIN KLEINES RATESPIEL ') ;    2000
  WRITELN        ('        SPIELEN ? ANTWORTEN SIE BITTE NUR MIT J') ;      2100
  WRITELN        ('        ODER MIT N ! ') ;                                2200
  WARTEN ;                 WRITELN ;     WRITELN ;                          2300
  REPEAT         READ      (Z) UNTIL Z <> ' ' ;                             2400
```

```
WHILE Z = 'J' DO                                                              2500
BEGIN                                                                         2600

  WRITELN    ('      DENKEN SIE SICH BITTE EINE BELIEBIGE ') ;                2700
  WRITELN    ('      GANZE ZAHL ! ') ;                                        2800
  WARTEN ;           WRITELN ;                                                2900
  WRITELN    ('      HABEN SIE SCHON EINE GEFUNDEN ! ') :                     3000
  WRITELN    ('      VERRATEN SIE SIE ABER NIEMANDEM ! ') ;                   3100
  WARTEN ;           WRITELN ;     WRITELN ;                                  3200
  WRITELN    ('      MULTIPLIZIEREN SIE IHRE ZAHL NUN MIT 3 . ') ;            3300
  WRITELN    ('      FUEGEN SIE NUN EINE 5 HINZU UND ') :                     3400
  WRITELN    ('      SUBTRAHIEREN SIE VOM BISHERIGEN ERGEBNIS ') ;            3500
  WRITELN    ('      DIE ZAHL 8 . ') ;                                        3600
  WARTEN ;           WARTEN ;     WRITELN ;  WRITELN :                        3700
  WRITELN    ('      SIND SIE SOWEIT ? ') ;                                   3800
  WRITELN    ('      JETZT NEHMEN SIE NOCH VON DEM REST ') :                  3900
  WRITELN    ('      DIE HAELFTE UND VERRATEN MIR. WAS SIE ') :               4000
  WRITELN    ('      HERAUSBEKOMMEN HABEN . ') :                              4100
  WRITELN    ('      DAS WAR DOCH SICHER NICHT ZU SCHWER ') :                 4200
  WRITELN    ('      FUER SIE ! ') ;                                          4300
  READ (Y) ;                                                                  4400
  X := 2 * Y/3 + 1 ;                                                          4500
  WRITELN    ('      DANKESCHOEN ! ') :                                       4600
  WRITELN :          WRITELN ;                                                4700
  WRITELN    ('      NUN KANN ICH IHNEN AUCH SCHON VERRATEN , ') :            4800
  WRITELN    ('      WELCHE ZAHL SIE SICH GEMERKT HATTEN . ') ;               4900
  WRITELN :          WRITELN ;                                                5000
  WRITELN    ('      ES WAR DIE ZAHL ' , X : 3 : 0) :                         5100
  WRITELN :                                                                   5200
  WRITELN    ('      HABE ICH RECHT ? ') ;                                    5300
  WRITELN :                                                                   5400
  WRITELN    ('      BIN ICH NICHT EIN SCHLAUES KERLCHEN ! ') ;               5500
  WARTEN ;           WARTEN ;     WRITELN ;     WRITELN ;                     5600
  WRITELN    ('      WOLLEN WIR NOCH EINMAL MITEINANDER SPIELEN ! ') :        5700
  WRITELN ;                                                                   5800
  WRITELN    ('      ANTWORTEN SIE BITTE NUR MIT J ODER MIT N . ') :          5900
  WRITELN ;          WRITELN ;                                                6000
  REPEAT     READ    (Z) UNTIL Z <> ' ' ;                                     6100
END ;        (*      ENDE DER WHILE – SCHLEIFE .                    *)        6200
  WRITELN    ('      SIE WOLLEN NICHT MEHR MIT MIR SPIELEN ! ') ;             6300
  WRITELN    ('      SCHADE. DENN MIR HAT DIE SACHE SEHR VIEL SPASZ ') ;      6400
  WRITELN    ('      BEREITET . ') :                                          6500
  WRITELN :                                                                   6600
  WRITELN    ('      ICH WUERDE MICH SEHR FREUEN . WENN SIE BALD ') ;         6700
  WRITELN    ('      WIEDER EINMAL MIT MIR SPIELEN WUERDEN . ') ;             6800
  WRITELN:                                                                    6900
  WRITELN    ('      ALSO BIS DAHIN : ') ;                                    7000
  WRITELN ;                                                                   7100
  WRITELN    ('          AUF WIEDERSPIELEN ! ') ;                             7200
  WRITELN ;          WRITELN ;     WRITELN ;     WRITELN ;                    7300
.END .       (*     ENDE DES SCHOENEN PROGRAMMS PASGAME 1 .  *)               7400
```

Das PASCAL-Programm wird beendet durch Programmzeile 7400 mit der Schlußanweisung END.

Autor:

*Prof. Dr.-Ing. Hans Kreul*
a.o. Prof. an der
Ingenieurhochschule Zittau
Abt. EDV und Rechentechnik

# »Intelligente« Eingaberoutine für Programme zur Realisierung einer komplexen Arithmetik

Die Darstellung komplexer Zahlen kann bekanntermaßen in unterschiedlicher Form erfolgen. Bild 1 kennzeichnet die Elemente einer komplexen Zahl in der Gaußschen Zahlenebene. Zum einen ist eine komplexe Zahl durch deren Koordinaten ($a$, $b$), zum anderen durch Radius und Winkel gekennzeichnet.

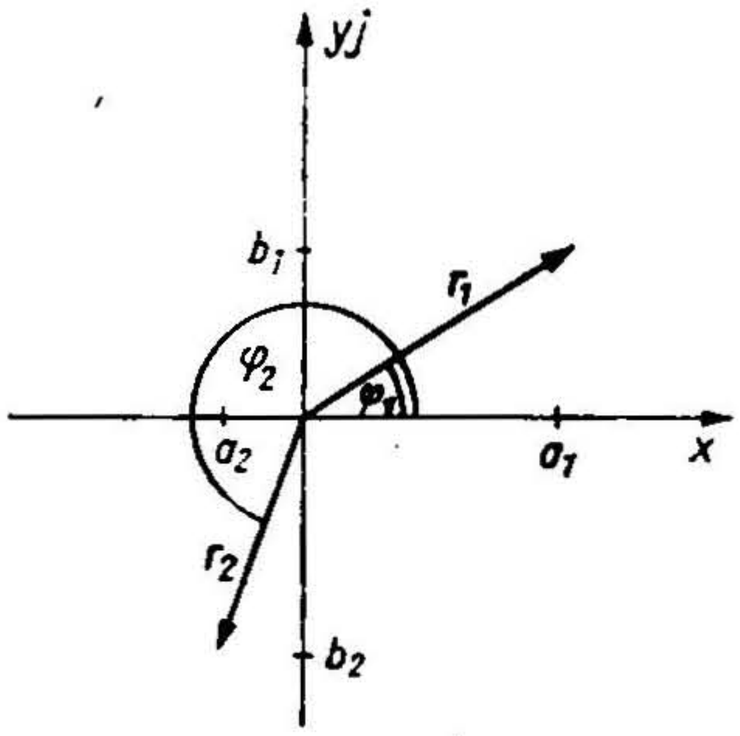



Bild 1. Komplexe Zahlen in der Gaußschen Zahlenebene

$$z^{\llcorner} = a + \mathrm{j}b$$

$$z^{\llcorner} = r\,(\cos\varphi + \mathrm{j}\sin\varphi) = r\exp \mathrm{j}\varphi$$

Beide Darstellungsformen einer komplexen Zahl sind äquivalent und ineinander überführbar. Arithmetische Operationen mit komplexen Zahlen lassen sich bei der richtigen Wahl der Darstellungsform sehr einfach durchführen. Um die Durchführung dieser Operationen weiter zu vereinfachen, sind eine Vielzahl von Programmen für Taschenrechner bzw. Kleincomputer bekannt. An dieser Stelle soll kein weiteres hinzugefügt werden. Ziel des Beitrags ist die Vorstellung einer »intelligenten« Eingaberoutine. Die »Intelligenz« der Eingabe besteht darin, daß die Herstellungsform der komplexen Zahl für die Eingabe **nicht** vorgeschrieben ist. Vom Programm werden die eingegebene Zeichenkette analysiert und die Bestimmungsgrößen der beiden Darstellungsformen der komplexen Zahl berechnet. Liegen dann Real- und Imaginärteil bzw. Radius und Winkel der komplexen Zahl vor, lassen sich die gewünschten arithmetischen Operationen problemlos anschließen.

Zur Demonstration des Vorgehens wurde ein einfaches BASIC-Programm für den an entsprechenden Bildungseinrichtungen und Industriestellen verbreiteten Mikrorechner MC-80 entworfen. Dem Demonstrationsanliegen kommen die eingeschränkten Möglichkeiten des implementierten BASIC-Dialekts entgegen. Das Listing des verwendeten MC-80-BASIC-Programms zeigt Bild 2.

Der Vereinbarungsteil weist keine Besonderheiten auf. Der mit einer Länge von 30 Zeichen vereinbarte String KOMP nimmt die zu analysierende Zahl auf. Nach dem Löschen des Bildschirms durch CLEAR erfolgt die Ausgabe von Hinweisen zum Eingabefor-

```
:    DEF UMRECHNUNG KOMPL. ZAHLEN
:    REAL A,B,C,D,E,F,G,H
:    REAL I,J,O,O,M,N,O,P,Q
:    REAL R,S,T,U,V,W,X,Y,Z
:    REAL REA,IMA,RAD,PHI,WINKEL
:    STRING KOMP(30),K,L,MERK
1    CLEAR
:    DPL 0,8'EINGABEFORMEN:'
:    DPL 2,0'+-REA+-IMAJ     ;REA=REALTEIL'
:    DPL 3,0'+-REA           ;IMA=IMAGINAERT.'
:    DPL 4,0'+-IMAJ          ;RAD=RADIUS'
:    DPL 5,0'+-RADEXP+-PHIJ  ;PHI=WINKEL'
:    DPL 7,25'(ENTER)'
:    WAIT
5    MERK='R'
:    CLEAR
10   DPL 0,0'WINKELANGABE IN GRD/RAD'(G/R):'MERK
:    DPL 1,12'(H=HILFESTELLUNG)'
:    DPL 0,30 :MERK=INP
:    IF MERK='H' THEN
:    GOTO 1
20   RAD=0 :IMA=0 :REA=0 :PHI=0 :I=-1 :Z=0
:    DPL 7,0'                                        '
:    DPL 6,0'EINGABE EINER KOMPLEXEN ZAHL:'
:    DPL 7,0 :KOMP=INP
30   I=I+1 :L=KOMP(I,1) :K=KOMP(I+1,1)
:    IF [K='+' OR K='-'] AND L<>'E' THEN  :REA=KOMP(0,I+1) :Z=I+1
:    GOTO 30
40   IF KOMP(I,3)='EXP' THEN RAD=KOMP(0,I) :I=I+2 :Z=I+1
:    IF RAD<=0 DO
:    GOTO 5
:    DOEND
:    GOTO 30
50   IF L='J' AND RAD=0 THEN IMA=KOMP(Z,Z-I)
:    GOTO 80
55   IF L='J' THEN PHI=KOMP(Z,Z-I)
:    GOTO 80
60   IF L=' ' THEN  :REA=KOMP(0,I)
:    GOTO 80
70   GOTO 30
80   IF RAD=0 THEN RAD=SQR(REA*REA+IMA*IMA)
:    LET PHI=FNATN(IMA/REA)
:    GOSUB 130
:    GOTO 110
90   IF MERK='G' DO PHI=PHI*2*PI/360
:    DOEND  :REA=RAD*COS(PHI) :IMA=RAD*SIN(PHI)
:    GOSUB 140
110  DPL 2,0'REALTEIL      ='REA
:    DPL 3,0'IMAGINAERTEIL='IMA
:    DPL 4,0'RADIUS        ='RAD
:    DPL 5,0'WINKEL(GRD)   ='PHI
:    IF MERK='R' THEN
:    DPL 5,7'RAD'
120  GOTO 10
130  IF REA<0 AND IMA<0 THEN PHI=PHI+PI
:    GOTO 140
132  IF REA>=0 AND IMA<0 THEN PHI=PHI+2*PI
:    GOTO 140
134  IF REA<0 AND IMA>=0 THEN PHI=PHI+PI
```

```
140 IF MERK$='G' THEN PHI=PHI*360/PI/2
160 RETURN
:   END
:   DEF FNATN(X)
:   REAL A,B,C,D,E,F,G,H,I,J,K
:   LET A=174.6554E-03 :B=3.709256 :C=6.76214 :D=-7.10676
:   LET E=3.316335 :F=-264.7686E-03 :G=1.448632 :H=1 :I=1
:   IF X<0 DO H=-1 :X=ABS(X)
:   DOEND
:   IF X>1 DO I=0 :X=1/X
:   DOEND  :J=X*X :K=X*(A+B/(J+C+D/(J+E+F/(J+G))))
:   IF I=0 DO K=PI/2-K
:   DOEND
:   FN=K*H
:   RETURN
:   FNEND
```

Bild 2. Listing der Eingaberoutine für komplexe Zahlen

```
EINGABEFORMEN:

+-REA+-IMAJ       ;REA=REALTEIL
+-REA             ;IMA=IMAGINAERT.
+-IMAJ            ;RAD=RADIUS
+-RADEXP+-PHIJ    ;PHI=WINKEL

                                (ENTER)
```

Bild 3. Hilfsmenü zur Erläuterung der Eingabeformate

mat der komplexen Zahl. Bild 3 zeigt eine entsprechende Bildschirmkopie. Durch die Betätigung der Taste ENTER wird die Abarbeitung des Programms mit dem Löschen des Bildschirms und der Ausgabe einer Eingabeaufforderung zur Festlegung der Winkelangabe (Grad- oder Bogenmaß) fortgesetzt. Durch Eingabe des Zeichens || werden erneut die Hinweise zum Eingabeformat ausgegeben.

Der Einbau solcher Hilfestellungen (»help functions«) sollte in dialogorientierten Programmen niemals fehlen!

Nach der Entscheidung für Grad- oder Bogenmaß bei der Winkelangabe kann die Eingabe einer komplexen Zahl in einem der vorgeschriebenen Eingabeformate erfolgen. Die Eingabeaufforderung und die eingegebene komplexe Zahl erscheinen an der Unterkante des Bildschirms. Real- und Imaginärteil sowie Radius und Winkel der komplexen Zahl lassen sich entsprechend der dem Rechner verständlichen Form als ganze (XXXXXX) oder gebrochene Zahl (XXX.XX bzw. XXX.XXEXX) eingeben.

Beginnend mit Marke 30, wird der Realteil des eingegebenen Strings gesucht. Der Realteil ist dann gefunden, wenn dem Vorzeichen im String kein »E« vorangestellt ist. Der Vergleich mit dem ersten in Bild 3 ausgegebenen Eingabeformat verdeutlicht den Sachverhalt. Ab Marke 40 wird im eingegebenen String die Zeichenfolge »EXP« gesucht. Die Stellen vor dieser Zeichenfolge bilden dann den Radius der komplexen Zahl. Das vierte Eingabeformat in Bild 3 verdeutlicht wieder die Aussage. Ein negativ eingegebener Radius wird als Fehler erkannt. Es erfolgt ein Rücksprung nach Marke 5, d.h., ein neuer Eingabezyklus wird gestartet.

Ab Marke 50 erfolgt die Suche nach dem Imaginärteil der komplexen Zahl. Als Imaginärteil gelten die Stellen vor dem die komplexe Zahl abschließenden »J« bis einschließlich Vorzeichen. Auf die geschilderte Weise ist der Imaginärteil im ersten und dritten Eingabeformat nach Bild 3 zu finden.

War der Radius nicht gleich Null, d.h., die komplexe Zahl wurde in Exponen-

tialform eingegeben (viertes Eingabeformat in Bild 3), dann werden ab Marke 55 die dem abschließenden »J« vorangehenden Stellen dem Winkel der komplexen Zahl zugeordnet.
Schließlich erfolgt ab Marke 60 die Suche nach reellen Zahlen (zweites Eingabeformat in Bild 3), deren abschließendes Zeichen ein Leerzeichen ist.
Nach der Analyse aller Eingabeformate schließt sich die Berechnung der fehlenden Bestimmungsgrößen an. Wurde die komplexe Zahl durch Eingabe von Real- und Imaginärteil bestimmt, ist die Bedingung RAD=Ø erfüllt, und es können ab Marke 80 Radius und Winkel berechnet werden. Die für die Winkelberechnung erforderliche Arcustangensfunktion ist im MC-80-BASIC nicht enthalten, weshalb der Funktionsverlauf der Arcustangensfunktion durch die in der Funktionsdefinition (DEF FNATN(X)) enthaltenen Beziehung einschließlich der angegebenen Konstanten approximiert wurde. Im Unterprogramm ab Marke 130 erfolgen Tests nach dem Quadranten der Gaussschen Zahlenebene, in dem die komplexe Zahl liegt, um ggf. erforderliche Korrekturen des Hauptwerts der Arcustangensfunktion vornehmen zu können. Ist die Angabe des Winkels in Gradmaß gewünscht worden, erfolgt bei Marke 140 noch die Umrechnung von Bogen- in Gradmaß.
War die Bedingung RAD=Ø nicht erfüllt, dann wird ab Marke 90 im Falle der Eingabe in Gradmaß eine Umrechnung von Grad- in Bogenmaß vorgenommen, um anschließend Real- und Imaginärteil der komplexen Zahl berechnen zu können. Ab Marke 140 schließt sich bei Erfordernis die Rückrechnung von Bogen- in Gradmaß an. Die Ausgabe der Ergebnisse kann nun beginnend bei Marke 110 in tabellarischer Form erfolgen. Die Bilder 4 bis 8 zeigen Bildschirmkopien für einige Beispiele.

```
WINKELANGABE IN GRD/RAD (G/R):R
          (H=HILFESTELLUNG)
REALTEIL      =-123.450
IMAGINAERTEIL=-543.210
RADIUS        = 557.061
WINKEL(RAD)   = 4.48892
EINGABE EINER KOMPLEXEN ZAHL:
-12345E-2-54321E-2J
```

Bild 4. Eingabe in der Form »-REA-IMAJ« Winkel in RAD

```
WINKELANGABE IN GRD/RAD (G/R):G
          (H=HILFESTELLUNG)
REALTEIL      =-123.450
IMAGINAERTEIL=-543.210
RADIUS        = 557.061
WINKEL(GRD)   = 257.196
EINGABE EINER KOMPLEXEN ZAHL:
-12345E-2-54321E-2J
```

Bild 5. Eingabe in der Form »-REA-IMAJ« Winkel in GRD

```
WINKELANGABE IN GRD/RAD (G/R):R
          (H=HILFESTELLUNG)
REALTEIL      =-4.53000
IMAGINAERTEIL= 0.00000
RADIUS        = 4.53000
WINKEL(RAD)   = 3.14159
EINGABE EINER KOMPLEXEN ZAHL:
-453E-2
```

Bild 6. Eingabe in der Form »-REA« Winkel in RAD

```
WINKELANGABE IN GRD/RAD (G/R):R
          (H=HILFESTELLUNG)
REALTEIL      = 0.00000
IMAGINAERTEIL= 24.0000
RADIUS        = 24.0000
WINKEL(RAD)   = 1.57080
EINGABE EINER KOMPLEXEN ZAHL:
24J
```

Bild 7. Eingabe in der Form »IMAJ« Winkel in RAD

```
WINKELANGABE IN GRD/RAD (G/R):G
          (H=HILFESTELLUNG)
REALTEIL      = 11.6924
IMAGINAERTEIL= 2.69941
RADIUS        = 12.0000
WINKEL(GRD)   = 13.0000
EINGABE EINER KOMPLEXEN ZAHL:
12EXP13J
```

Bild 8. Eingabe in der Form »RADEXPPHIJ« Winkel in GRD

Im vorliegenden Beitrag sollte der Versuch unternommen werden, eine »intelligente« Eingaberoutine für Programme zur Realisierung einer komplexen Arithmetik mit den diesbezüglich eingeschränkten Möglichkeiten des im Mikrorechner MC-80 implementierten BASIC-Dialektes zu schaffen. Dialogunterstützend wurde eine Hilfsfunktion in dem Sinne geschaffen, daß jederzeit nach einem entsprechenden Aufruf die die Eingabeformate beschreibenden Hinweise auf dem Bildschirm ausgegeben werden. Auf diese Weise müssen bei der Eingabe komplexer Zahlen keine bestimmten Darstellungsformen vorgegeben werden. Hierzu erforderliche Umrechnungen von einer Darstellungsform in die andere entfallen damit für den Nutzer eines dergestalt ausgerüsteten Programms zur Realisierung einer komplexen Arithmetik.

*Ulrich Hähnel*
Mitarbeiter Forschung und Entwicklung
VEB Pentacon Dresden

*Dr. Claus Kühnel*
Wissenschaftlicher Mitarbeiter im Zentralinstitut des Sportmedizinischen Dienstes
Kreischa

## Lösungen zu »Beherrschen Sie BASIC?«

*Zu Aufgabe 1*

Die Laufvariable I wird in der Anweisung 3Ø stets um 1 vermindert, während sie durch die Laufanweisung 2Ø immer um 1 erhöht wird. Damit behält sie ihren Wert bei und kann daher nie den Endwert I = 1ØØ erreichen. Es werden also ständig nur Nullen ausgedruckt, das Programm kann nur durch Eingriff von außen (z.B. Drücken der STOP-Taste) zum Halten gebracht werden.

*Zu Aufgabe 2*

Die Anweisung 3Ø erzeugt ganzzahlige Zufallszahlen, die zwischen 1 und 35 liegen. (Frage: Warum wurde an Stelle der Anweisung 3Ø nicht
L = INT(35∗RND(K)) geschrieben?)
Es entstehen also 5 Zufallszahlen aus dem Bereich 1 bis 35, und damit könnte das Programm zum Ausfüllen eines Tippscheines für die Wettart 5 aus 35 verwendet werden.
Bei dem Programm kann es allerdings vorkommen, daß eine der zufällig erzeugten 5 Zahlen in dem Fünfertip mehrfach auftritt. Dann müßte man das Programm eben noch einmal aufrufen. Oder – und das wäre besser – man müßte sich überlegen, wie man durch zusätzliche Anweisungen vermeiden kann, daß eine Zahl mehrfach auftritt.

*Zu Aufgabe 3*

Selbstverständlich kann er die Variablen auseinander halten, denn er ist ja ein kluges Kerlchen!
Das durch das Programm erzeugte Druckbild sieht folgendermaßen aus:

| | | | |
|---|---|---|---|
| 1 | HANS | 6 | A |
| 2 | HANS | 7 | B |
| 3 | HANS | 8 | C |
| 4 | HANS | 9 | D |
| 5 | HANS | 1Ø | E |
| 6 | HANS | 11 | F |
| 7 | HANS | 12 | G |
| 8 | HANS | 13 | H |
| 9 | HANS | 14 | I |
| 1Ø | HANS | 15 | J |

Das vorgeführte Programm ist jedoch ein Musterbeispiel dafür, wie man **nicht** programmieren sollte. Es ist dringend anzuraten, für *unterschiedliche Variablen* auch *unterschiedliche Bezeichnungen* (Namen) zu verwenden.

Autor:
*Prof. Dr.-Ing. Hans Kreul*
a. o. Professor an der Ingenieurhochschule Zittau

# Programm zur Berechnung von Kennwerten einfacher Verstärkerstufen mit bipolaren Transistoren

Die Dimensionierung von Verstärkerstufen geschieht häufig iterativ, d.h., ausgehend von einer ersten Dimensionierung werden die Kennwerte der Transistorstufe berechnet, diese mit den Vorgaben verglichen und ggf. entsprechende Parameter verändert. Die Eigenschaften einer Verstärkerstufe lassen sich durch die Kennwerte Eingangswiderstand, Ausgangswiderstand und Verstärkungsfaktor beschreiben. Der Arbeitspunkt des Transistors ist von weiterem Interesse.

Das im folgenden beschriebene Programm wurde für den Mikrorechner MC-80 in BASIC entworfen und erlaubt die Berechnung der o.g. Kennwerte für die

- Emitterschaltung mit Spannungsgegenkopplung,
- Emitterschaltung mit Stromgegenkopplung und den
- Emitterfolger.

## Grundlagen

Die zu betrachtenden Schaltungen sind in den Bildern 1 bis 3 dargestellt. Bild 1 zeigt eine Verstärkerstufe in Emitterschaltung mit Spannungsgegenkopplung. In Bild 2 ist die Emitterschaltung mit Stromgegenkopplung einschließlich des zur Arbeitspunkteinstellung erforderlichen Basisspannungsteilers angegeben. Der Emitterfolger nach Bild 3 unterscheidet sich von der Emitterschaltung mit Stromgegenkopplung nur hinsichtlich des Abgriffs des Ausgangssignals.

Unter Verwendung einer NF-Kleinsi-

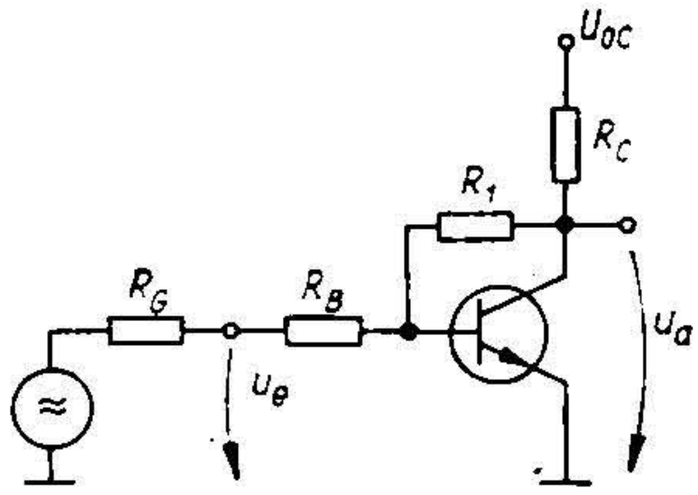



Bild 1. Emitterschaltung mit Spannungsgegenkopplung

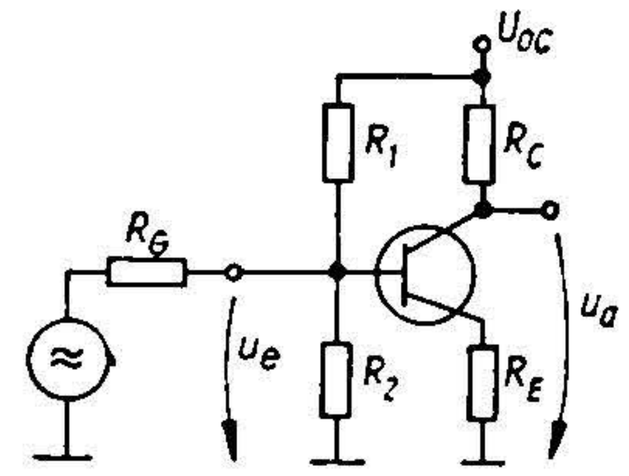



Bild 2. Emitterschaltung mit Stromgegenkopplung

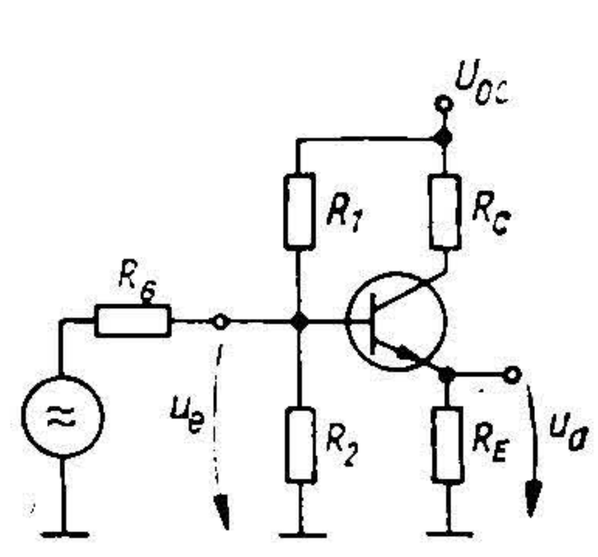



Bild 3. Emitterfolger

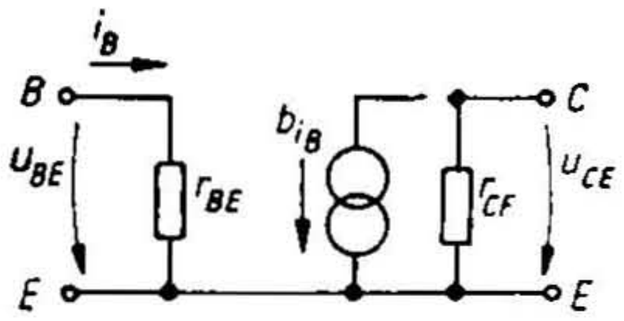



Bild 4. NF-Kleinsignal-Ersatzschaltbild des Bipolartransistors

gnal-Ersatzschaltung für den Bipolartransistor lassen sich die Kleinsignalkenngrößen berechnen. Für überschlägige Berechnungen sind bereits mit dem in Bild 4 gezeigten NF-Kleinsignal-Ersatzschaltbild hinreichend genaue Ergebnisse zu erreichen. Im folgenden sollen unter Verzicht auf die entsprechende Herleitung die zur Berechnung der Kennwerte benutzten Bezeichnungen angegeben werden:

Emitterschaltung mit Spannungsgegenkopplung:

$$\frac{1}{v'} \approx -\frac{R_B}{R_1} + \frac{1}{v} \tag{1}$$

mit

$$v \approx -\frac{b}{r_{BE}} (R_C \| r_{CE} \| R_1) \tag{2}$$

$$r_e' \approx R_B + \frac{R_1}{1-v} \| r_{BE} \tag{3}$$

$$r_a' \approx \frac{1}{b} \left( R_1 + r_{BE} + \frac{R_1 r_{BE}}{R_1 + R_C} \right) \| \| R_C \| r_{CE} \| R_1 \tag{4}$$

Emitterschaltung mit Stromgegenkopplung:

$$\frac{1}{v'} \approx -\frac{R_E}{R_C} + \frac{1}{v} \tag{5}$$

$$r_e' \approx (r_{BE} + b R_E) \| R_1 \| R_2 \tag{6}$$

$$r_a' \approx R_C \| \left( r_{CE} \left( 1 + \frac{b R_E}{r_{BE} + R_E + R_C} \right) \right) \tag{7}$$

Emitterfolger:

$$\frac{1}{v'} \approx 1 + \frac{r_{BE}}{b R_E} \tag{8}$$

$$r_e' \approx (r_{BE} + b R_E) \| R_1 \| R_2 \tag{9}$$

$$r_a' \approx R_E \| \frac{r_{BE} + R_C}{b} \tag{10}$$

In den Gln. (1) bis (10) wurden die Kennwerte der gegengekoppelten Schaltungen durch ein »'« gekennzeichnet. Die Größen sind durch die Dimensionierung bzw. die Transistorkennwerte vorgegeben. Abgeleitet aus den Kennliniengleichungen des Bipolartransistors gilt für den Basis-Emitterwiderstand $r_{BE}$ die Beziehung

$$r_{BE} = \frac{b U_T}{I_C} \text{ mit } U_T = \frac{k T}{e} \tag{11}$$

$k$ bezeichnet die BOLTZMANN-Konstante und $e$ die Elementarladung. $T$ steht für die absolute Temperatur.

Für den Kollektor-Emitterwiderstand $r_{CE}$ wurde die für die Praxis ausreichende Näherung

$$r_{CE} \approx 200000\, r_{BE}/b^2 \tag{12}$$

verwendet. Der Stromverstärkungsfaktor $b$ wird dem verwendeten Transistor entsprechend vorgegeben.

Zur Überprüfung der Gültigkeit der vorgegebenen Dimensionierung ist die Berechnung des Kollektorstroms $I_C$, sowie der Kollektor- und Emitterspannungen $U_C$ und $U_E$ erforderlich.

Mit der Gl.

$$U_{BE} = U_T \ln \frac{I_C}{I_S} \tag{13}$$

erfolgt eine statische Analyse durch iterative Lösung der Beziehung

$$0 = \frac{R_2}{R_1 + R_2} U_{OC} - U_{BE} - \left( R_E + \left( \frac{R_1}{b} + R_C \right) \frac{R_2}{R_1 + R_2} \right) \times I_S \exp \frac{U_{BE}}{U_T}. \tag{14}$$

Der Strom $I_S$ kennzeichnet den Sättigungsstrom des Transistors. Er wurde an einem integrierten Transistorarray B 340 (HFO) ermittelt und betrug $I_S = 0{,}7$ fA. Diese Größe ist abhängig von der Geometrie des Transistors und ist damit typspezifisch. Mit der Span-

nung $U_{BE}$ im Arbeitspunkt lassen sich dann auf einfache Weise Kollektorstrom sowie Kollektor- und Emitterspannung berechnen.

**Handhabung des Programms**

Nach dem Start des Programms erscheint auf dem Bildschirm ein Menü, welches die Auswahl der gewünschten Schaltungsart ermöglicht. Der Dialog mit dem Rechner wird durch eine Reihe von Druckerlisten dokumentiert. Die Emitterschaltung mit Spannungsgegenkopplung (Liste 1) wird durch die Eingabe von »SP« auf die Frage »Typ ?« ausgewählt. Bei der Eingabe der Widerstandswerte ist bei Nichtvorhandensein eines Widerstands ein hoher Wert (hier $R_2 = 100\,\mathrm{M\Omega}$) und bei Kurzschluß ein sehr kleiner Wert (hier $R_E = 1\,\Omega$) zu verwenden.

```
SCHALTUNGSAUSWAHL:

 ES M. SPANNUNGSGEGENKOPPL.: SP
 ES M. STROMGEGENKOPPLUNG  : ST
 EMITTERFOLGER             : EF

      TYP?   SP


 R1 = 82.0000E+03
 R2 = 100.000E+06
 RC = 1.00000E+03
 RE = 1.00000
 RG = 10.0000
  B = 100.000
 UOC= 10.0000
 UOB=UC? (Y=1,N=0)   1


ES M. SPANNUNGSGEGENKOPPLUNG

    RB=?   10.0000E+03


ES M. SPANNUNGSGEGENKOPPLUNG

   IC   = 5.10152E-03 03
   UC   = 4.89847
   UE   = 5.10152E-03
   V    =-7.83638
   REIN= 10.2422E+03
   RA   = 8.91395
```

Liste 1

```
SCHALTUNGSAUSWAHL:

 ES M. SPANNUNGSGEGENKOPPL.: SP
 ES M. STROMGEGENKOPPLUNG  : ST
 EMITTERFOLGER             : EF

      TYP?   ST


 R1 = 10.0000E+03
 R2 = 10.0000E+03
 RC = 1.00000E+03
 RE = 1.00000E+03
 RG = 10.0000
  B = 100.000
 UOC= 10.0000
 UOB=UC? (Y=1,N=0)   0


ES M. STROMGEGENKOPPLUNG

   IC   = 4.09136E-03
   UC   = 5.90864
   UE   = 4.09136
   V    =-993.192E-03
   REIN= 4.76334E+03
   RA   = 998.728
```

Liste 2

Die Beantwortung der Frage »UOB = UC? (Y=1, N=Ø)« kennzeichnet den Anschluß des Widerstands $R_1$. Eine Eingabe von »1« bedeutet, daß der Widerstand $R_1$ an den Kollektor des Transistors geführt wird (SP), dementsprechend kennzeichnet die Eingabe von »Ø« den Anschluß an die Betriebsspannung $U_{OC}$ (ST, EF). Schließlich muß noch die Eingabe des Basisvorwiderstands erfolgen, bevor die Ergebnisse zur Anzeige gebracht werden.
Liste 2 zeigt die Ergebnisse einer Emitterschaltung mit Stromgegenkopplung. Hier sind alle Widerstände vorhanden, und der zum Basis-Spannungsteiler gehörende Widerstand $R_1$ liegt an der Betriebsspannung $U_{OC}$. Die Berechnung einer Emitterschaltung mit Stromgegenkopplung bei veränderter Dimensionierung zeigt Liste 3. Hier zeigt die statische Analyse, daß auf Grund der Bedingung $U_E$ größer $U_C$ eine Neudimensionierung vorgenommen werden muß. Die Berechnungen

```
SCHALTUNGSAUSWAHL:

 ES M. SPANNUNGSGEGENKOPPL.: SP
 ES M. STROMGEGENKOPPLUNG  : ST
 EMITTERFOLGER             : EF

      TYP?   ST


 R1 = 1.00000E+03
 R2 = 1.00000E+03
 RC = 10.0000E+03
 RE = 4.00000E+03
 RG = 10.0000
  B = 100.000
 UOC= 10.0000
 UOB=UC? (Y=1,N=0)   0


UE>UC, NEUDIMENSIONIERUNG!
```

Liste 3

```
SCHALTUNGSAUSWAHL:

 ES M. SPANNUNGSGEGENKOPPL.: SP
 ES M. STROMGEGENKOPPLUNG  : ST
 EMITTERFOLGER             : EF

      TYP?   EF


 R1 = 10.0000E+03
 R2 = 10.0000E+03
 RC = 1.00000
 RE = 1.00000E+03
 RG = 10.0000
  B = 100.000
 UOC= 10.0000
 UOB=UC? (Y=1,N=0)   0


EMITTERFOLGER

   IC   = 4.09136E-03
   UC   = 9.99591
   UE   = 4.09136
   V    = 993.685E-03
   REIN= 4.76334E+03
   RA   = 6.41346
```

Liste 4

zum Emitterfolger sind mit Liste 4 gegeben. Der Kollektorwiderstand wurde aus den o. a. Gründen mit $R_C = 1\,\Omega$ eingegeben.
Für Interessenten soll abschließend eine Programmliste die Nachnutzung ermöglichen.

## Schlußbemerkung

Nach Meinung der Verfasser läßt sich das vorgestellte Programm vorteilhaft dort einsetzen, wo ohne umständliche Rechenarbeit die Zusammenhänge zwischen Schaltungsdimensionierung und Schaltungseigenschaften bei einfachen Verstärkerstufen verdeutlicht werden sollen.
Der Abdruck der Programmliste soll auch Anpassungen an andere BASIC-Dialekte ermöglichen (s. S. 64).
Andere Grundschaltungen, wie beispielsweise die Basisschaltung, wurden nicht berücksichtigt, da das gewählte Ersatzschaltbild für den Bipolartransistor für die betreffenden Frequenzen unzureichend ist. Im Bereich mittlerer bzw. hoher Frequenzen ist die Frequenzabhängigkeit der Transistorkennwerte zu berücksichtigen, was jedoch kompliziertere Beschreibungsformen zur Folge hätte.

*Ulrich Hähnel*
Mitarbeiter Forschung und Entwicklung
VEB Pentacon Dresden

*Dr. Claus Kühnel*
Wissenschaftlicher Mitarbeiter im Zentralinstitut des Sportmedizinischen Dienstes
Kreischa

```
:    DEF TRANS.-SCHALTUNGSBER.
:    REAL A,B,C,D,E,F,G,H
:    REAL I,J,K,L,M,N,O,P,Q
:    REAL R,S,T,U,V,W,X,Y,Z
:    REAL R1,R2,RC,RE,RG,REIN,RB
:    REAL RBE,RCE,RA,RS,UOC,UBE
:    REAL UE,UC,DUBE,HILF,R10
:    REAL RCS,VO,IC
:    STRING TYP(2),UBC(1)
10   DPL 0,0'SCHALTUNGSAUSWAHL:'
:    DPL 2,1'ES M. SPANNUNGSGEGENKOPPL.: SP'
:    DPL 3,1'ES M. STROMGEGENKOPPLUNG  : ST'
:    DPL 4,1'EMITTERFOLGER'TAB(13)': EF'
:    DPL 6,6'TYP?   '
12   LET TYP=INP
:    CLEAR
14   DPL 0,1'R1 ='
:    LET R1=INP
:    DPL 0,5R1
:    DPL 1,1'R2 ='
:    LET R2=INP
:    DPL 1,5R2
:    DPL 2,1'RC ='
:    LET RC=INP
:    DPL 2,5RC
:    DPL 3,1'RE ='
:    LET RE=INP
:    DPL 3,5RE
:    DPL 4,1'RG ='
:    LET RG=INP
:    DPL 4,5RG
:    DPL 5,2'B ='
:    LET B=INP
:    DPL 5,5B
:    DPL 6,1'UOC='
:    LET UOC=INP
:    DPL 6,5UOC
:    DPL 7,1'UOB=UC? (Y=1,N=0)   '
:    LET HILF=INP
16   LET UBE=0
:    LET DUBE=2
:    LET RCS=RC*HILF
:    LET RS=R2/(R1+R2)
:    LET R10=-8.699988E-15*(RS*(R1/B+RCS)+RE)
20   LET DUBE=DUBE/2
:    CLEAR
:    LET N=(EXP(UBE/26.00001E-03)*R10)+UOC*RS-UBE
:    IF N>=0 THEN
:    GOTO 30
25   LET UBE=UBE-DUBE
:    GOTO 40
30   LET UBE=UBE+DUBE
40   IF DUBE-100E-06>=0 THEN
:    GOTO 20
42   LET IC=8.699978E-15*EXP(UBE/26.00001E-03)
:    LET RBE=B/IC*26.00002E-03
:    LET RCE=RBE*200E+03/B/B
:    LET UE=RE*IC
:    LET UC=UOC-IC*RC
44   CLEAR
:    IF TYP='SP' THEN
:    GOTO 70
45   IF TYP='ST' THEN
:    GOTO 60
```

```
46  IF TYP='EF' THEN
:   GOTO 50
48  GOTO 10
50  DPL 0,0'EMITTERFOLGER'
:   LET REIN=1/(1/(B*RE+RBE)+1/R1+1/R2)
:   LET V=1/(RBE/B/RE+1)
:   LET RA=1/(1/((RBE+RG)/B)+1/RE)
:   GOTO 80
60  DPL 0,0'ES M. STROMGEGENKOPPLUNG'
:   LET V=1/(-1/(1/(1/RC+1/RCE)*B/RBE)-RE/RC)
:   LET REIN=1/(1/(B*RE+RBE)+1/R1+1/R2)
:   LET RA=1/(1/((1/(RBE+RE+RG)*B*RE+1)*RCE)+1/RC)
:   GOTO 80
70  DPL 0,0'ES M. SPANNUNGSGEGENKOPPLUNG'
:   DPL 2,4'RB=?  '
:   LET RB=INP
:   DPL 2,10RB
:   LET V0=-1/(1/RC+1/RCE+1/R1)*B/RBE
:   LET V=1/(1/V0-RB/R1)
:   LET REIN=1/((1-V0)/R1+1/RBE)+RB
:   LET RA=1/(1/R1+1/(R1*RBE/(RB+RG)+RBE+R1)+1/RC+1/RCE)/B
80  IF UC-R10)=0 THEN
:   GOTO 90
85  CLEAR
:   DPL 0,0'UE>UC, NEUDIMENSIONIERUNG!'
:   WAIT
:   CLEAR
:   GOTO 10
90  DPL 2,2'IC   ='IC
:   DPL 3,2'UC   ='UC
:   DPL 4,2'UE   ='UE
:   DPL 5,2'V    ='V
:   DPL 6,2'REIN='REIN
:   DPL 7,2'RA   ='RA
:   WAIT
:   CLEAR
:   GOTO 10
```

ISBN 3-343-00226-7



1. Auflage
Lizenznummer 114-210/2/86
LSV 1083
Verlagslektor: Helga Fago
Gestaltung: Lothar Gabler
Printed in GDR

Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328

Redaktionsschluß: 15. 9. 1986
Bestellnummer: 547 1324
00780

Kleinstrechner-TIPS / Hrsg. von
Hans Kreul u.a. – Leipzig: Fachbuchverl.,
H. 6. – 1. Aufl. – 1987. – 64 S.: 25 Bild.

Anschrift des Verlages:

VEB Fachbuchverlag
PSF 67

DDR – Leipzig
7031

Die Broschürenreihe
KLEINSTRECHNER-TIPS
behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00226-7