Die Anwendung programmierbarer Taschenrechner im Unterricht der Sekundarstufe II sollte sich vornehmlich an bestehenden Lernzielen orientieren. Darüberhinaus lassen sich aber Inhalte vermitteln, die ohne den Einsatz von Computern nicht zugänglich wären. Wie dabei der programmierbare Taschenrechner als Rechenhilfsmittel und methodisches Hilfsmittel einsetzbar ist, wird an Beispielen für die Behandlung der Kombinatorik, der Verteilungsfunktionen und der stochastischen Simulation gezeigt.

Volkhart Lehmann

Anwendung programmierbarer Taschenrechner in der Wahrscheinlichkeitsrechnung und Statistik – Sekundarstufe II

1 Kombinatorik

Fakultäten lassen sich rekursiv definieren durch $0! = 1$ und $n! = (n-1)! \cdot n$. Die Erstellung eines Programms zur Berechnung von Fakultäten im Unterricht ist aus zwei Gründen besonders interessant: Hier ist eine Möglichkeit zur Einführung in das *Programmieren von Schleifen* gegeben. Programmiert man entsprechend dem **Struktogramm** in **Fig. 1**, so ist der Sonderfall $n = 0$ mit berücksichtigt, was nicht der Fall wäre, wenn die Abbruchbedingung $i = 0$ an den Schluß der Schleife gesetzt würde.

Man kann hier auf die Sprachelemente höherer Programmiersprachen **Repeat-until** (Abfrage am Schluß) und **While-Schleife** (Abfrage am Anfang) aufmerksam machen.

Der Begriff des Unterprogramms läßt sich bei den k-Permutationen (**Fig. 2**) über die Definition $P_k(n) = \frac{n!}{(n-k)!}$ und bei den Binomialkoeffizienten über die Definition $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ einführen. Ein solches Programm ist aber sehr langsam. Zur Berechnung von $\binom{49}{6}$ werden auf dem TI-58 ca. 42 Sekunden benötigt. Zudem ist der Eingabebereich eines solchen Programms stark eingeschränkt, da für $n > 69$ **Overflow** auftritt.

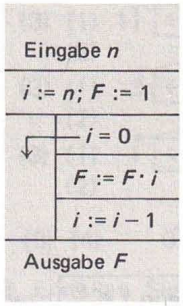

Fig. 1 n-Fakultät

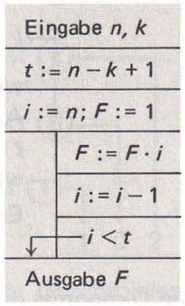

Fig. 2 k-Permutationen

Struktogramm: Die graphische Darstellungsmethode für Programmabläufe in Form eines Struktogramms setzt sich immer mehr durch. Vorgestellt und erklärt wurde diese Methode im „Jahrbuch '80" (*W. Schneider:* Graphische Darstellung von Programmabläufen).

Repeat-until: „Wiederhole, bis" eine Bedingung erfüllt ist. Wichtige Form der Programmierung von Schleifen. Dabei wird ein Programmstück so oft wiederholt, bis z. B. ein Vergleichswert null geworden ist.

While-Schleife: *While* heißt „während". Gemeint ist hier, daß eine Programmschleife so lange durchlaufen wird, wie eine Bedingung erfüllt ist (wenn Bedingung erfüllt, durchlaufe Schleife).

Overflow: Überlauf, Bereichsüberschreitung.

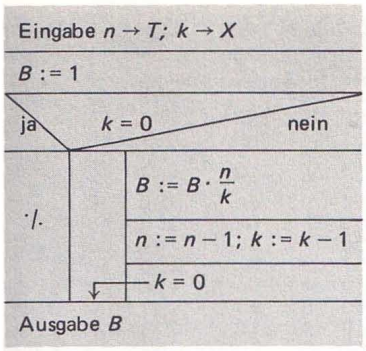

Fig. 3 Binomialkoeffizienten

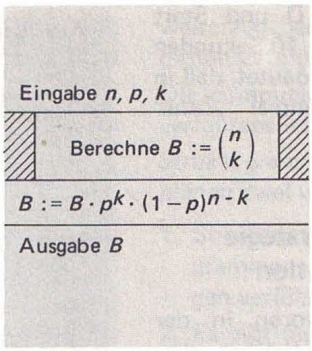

Fig. 4 Binomialwahrscheinlichkeit

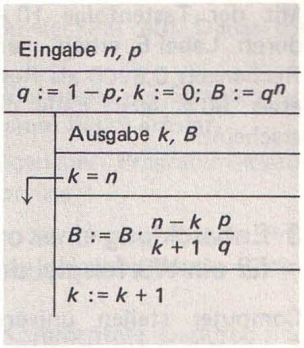

Fig. 5 Binomialverteilung

Es liegt daher nahe, nach einer Rekursionsformel zu suchen und entsprechend dem hier angegebenen Struktogramm (**Fig. 3**) zu programmieren.

2 Verteilungsfunktionen

Einen wichtigen Abschnitt im Stochastikunterricht bildet wegen der Anwendungsbezogenheit das Testen von Hypothesen. Dabei wird wegen seiner Einfachheit meist das Binomialmodell zugrunde gelegt, so daß die Anwendung programmierbarer Taschenrechner in diesem Themenbereich sinnvoll ist. Bei der Berechnung der Binomialwahrscheinlichkeit (**Fig. 4**) kann man auf das bereits entwickelte Unterprogramm „Binomialkoeffizienten" zurückgreifen.

Den Einfluß der Parameter n und p wird man mit dem wiederum rekursiv arbeitenden Programm „Binomialverteilung" (**Fig. 5**) studieren. In Gruppenarbeit lassen sich durch Variation der Parameter n und p für verschiedene Verteilungsfunktionen Wertetabellen und zugehörige **Histogramme** erstellen. An dieser Stelle ist auch ein Vergleich mit der hypergeometrischen Verteilung (Ziehen ohne Zurücklegen) und der Poissonverteilung möglich.

Für das Testen von Hypothesen nach dem Binomialmodell ist jedoch ein Programm wichtiger, das gestattet, die summierte Binomialwahrscheinlichkeit $B_{n;p}\,(i \leqslant X \leqslant j)$ zu berechnen (**Fig. 6**). Ein solches Programm macht unabhängig von den in Tabellen vorhandenen Werten und ist für den TI-58/59 hier angegeben, zumal ein solches Programm nicht in dem Standardmodul verfügbar ist (**Fig. 7**). Die Eingabe der Werte erfolgt durch *Programmadreßlabel:*

n A *p* B *i* C *k* D

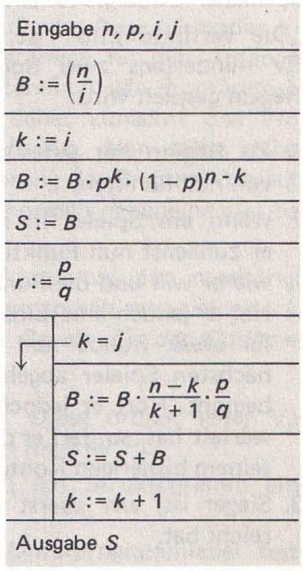

Fig. 6 $B_{n;p}\,(i \leqslant X \leqslant j)$

und der Start des Programms durch **Label** E. Wir testen das Programm an folgendem *Beispiel:*

Wie groß ist die Wahrscheinlichkeit, daß bei zehnmaligem Münzwurf die Anzahl der Einsen um höchstens 2 vom Erwartungswert 5 abweicht?

Histogramm: Graphische Darstellung einer Häufigkeitsverteilung. Besonderheit: Die unabhängige Veränderliche ist oft in gleich große Abschnitte unterteilt (Treppendarstellung).

Label: Marke, Kennzeichnung. In der Programmiertechnik bezeichnet man damit Abkürzungen oder Symbole, die bestimmte Speicher- oder Programmadressen darstellen. Daher stammt auch der Ausdruck „Programmadreßlabel".

Mit der Tastenfolge 10 A 0.5 B 3 C 7 D und Start durch Label E erhält man nach etwa 10 Sekunden Rechenzeit 0,8906 als Resultat. Das bedeutet, daß in etwa 90% aller Fälle drei- bis siebenmal „Zahl" erscheint.

3 Entwicklung einer optimalen Strategie für ein Würfelspiel durch Simulation

Computer stellen universelle Simulatoren in der Stochastik und in den Naturwissenschaften dar. Beim Einsatz von programmierbaren Taschenrechnern im Unterricht wird man daher auf diese Anwendungsmöglichkeit nicht verzichten. Als Beispiel zeigen wir hier, wie man durch Simulation zu einer optimalen Strategie für ein Würfelspiel kommt.

„Die verflixte Eins" ist ein amüsantes Würfelspiel für mindestens zwei Spieler, das nach folgenden Regeln gespielt wird:

1. Zu Beginn hat jeder Spieler die Gesamtsumme von null Punkten.
2. Wenn ein Spieler in seiner Runde anfängt, hat er zunächst null Punkte. Er darf so lange würfeln, wie er will und die Punktzahlen zusammenzählen. Hat er jedoch eine Eins gewürfelt, so bekommt er für *diese* Runde null Punkte und muß an den nächsten Spieler abgeben, der dann seine Runde beginnt. Hört er jedoch auf, ehe er eine Eins gewürfelt hat, so darf er die bisher erreichten Punkte seinem bisherigen Kontostand zuschlagen.
3. Sieger ist, wer zuerst 101 oder mehr Punkte erreicht hat.

Dieses Spiel läßt sich unter Verwendung des PTR als „Elektronischem Würfel" durchführen. Dabei kann man zum Beispiel das folgende Programm benutzen, das durch Drücken der Taste A aufgerufen wird (**Fig. 8**). Vor Beginn sollte der Zufallsgenerator durch Eingabe einer Zufallszahl nach R 00 vorbereitet werden.

Für den Spieler erhebt sich nach jedem Würfeln die Frage, ob er noch einmal würfeln soll oder ob es besser ist, aufzuhören und sich die erreichte Punktzahl gutschreiben zu lassen. Mit steigender Wurfzahl wächst in jeder Runde die Gefahr, eine Eins zu würfeln. Wir wenden daher die folgende Strategie an:

Man führt in jeder Runde eine bestimmte Anzahl von Würfen durch und hört dann auf. Doch für welche Wurfzahl je Runde erhält man nach diesem Verfahren eine optimale Strategie?

ADR	Code	Taste	ADR	Code	Taste	ADR	Code	Taste
000	76	LBL	048	02	02	096	43	RCL
001	11	A	049	01	1	097	04	04
002	42	STO	050	22	INV	098	95	=
003	03	03	051	44	SUM	099	42	STO
004	91	R/S	052	01	01	100	08	08
005	76	LBL	053	97	DSZ	101	76	LBL
006	12	B	054	00	00	102	32	X:T
007	42	STO	055	22	INV	103	43	RCL
008	04	04	056	76	LBL	104	06	06
009	91	R/S	057	23	LNX	105	32	X:T
010	76	LBL	058	43	RCL	106	43	RCL
011	13	C	059	05	05	107	00	00
012	42	STO	060	42	STO	108	67	EQ
013	05	05	061	00	00	109	33	X²
014	91	R/S	062	43	RCL	110	53	(
015	76	LBL	063	02	02	111	43	RCL
016	14	D	064	65	×	112	03	03
017	42	STO	065	43	RCL	113	75	-
018	06	06	066	04	04	114	43	RCL
019	91	R/S	067	45	Y^X	115	00	00
020	76	LBL	068	43	RCL	116	54	)
021	15	E	069	05	05	117	55	÷
022	43	RCL	070	65	×	118	53	(
023	03	03	071	53	(	119	43	RCL
024	42	STO	072	01	1	120	00	00
025	01	01	073	75	-	121	85	+
026	43	RCL	074	43	RCL	122	01	1
027	05	05	075	04	04	123	54	)
028	42	STO	076	54	)	124	65	×
029	00	00	077	45	Y^X	125	43	RCL
030	01	1	078	53	(	126	08	08
031	42	STO	079	43	RCL	127	95	=
032	02	02	080	03	03	128	49	PRD
033	43	RCL	081	75	-	129	02	02
034	00	00	082	43	RCL	130	43	RCL
035	29	CP	083	05	05	131	02	02
036	67	EQ	084	54	)	132	44	SUM
037	23	LNX	085	95	=	133	07	07
038	76	LBL	086	42	STO	134	01	1
039	22	INV	087	02	02	135	44	SUM
040	43	RCL	088	42	STO	136	00	00
041	01	01	089	07	07	137	61	GTO
042	49	PRD	090	43	RCL	138	32	X:T
043	02	02	091	04	04	139	76	LBL
044	43	RCL	092	55	÷	140	33	X²
045	00	00	093	53	(	141	43	RCL
046	22	INV	094	01	1	142	07	07
047	49	PRD	095	75	-	143	91	R/S

Fig. 7 Summierte Binomialwahrscheinlichkeit

ADR	Code	Taste	ADR	Code	Taste	ADR	Code	Taste
000	76	LBL	007	45	Y^X	014	65	×
001	11	A	008	05	5	015	06	6
002	43	RCL	009	95	=	016	85	+
003	00	00	010	22	INV	017	01	1
004	85	+	011	59	INT	018	95	=
005	89	π	012	42	STO	019	59	INT
006	95	=	013	00	00	020	92	RTN

Fig. 8 Elektronischer Würfel

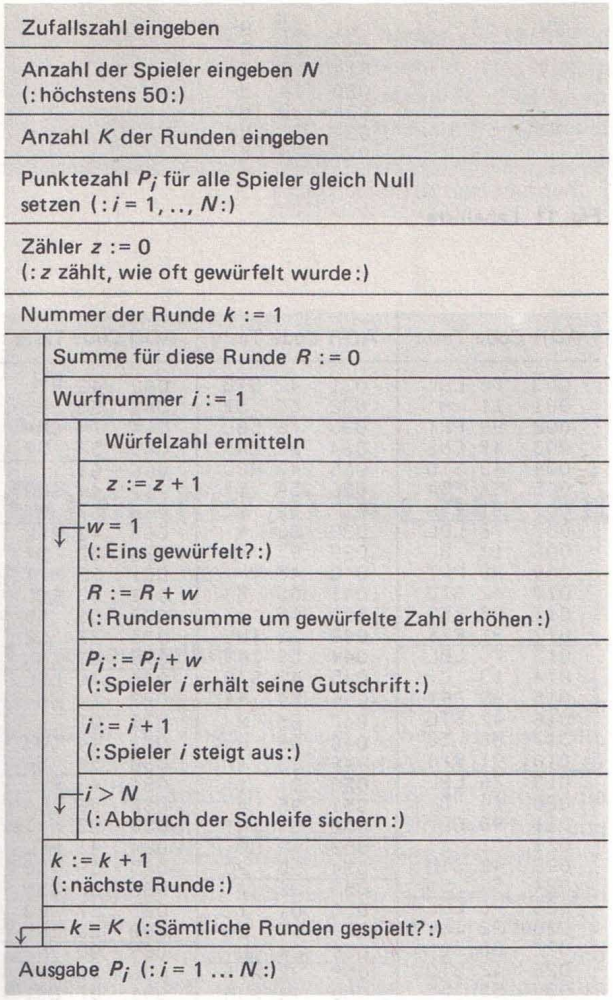

Fig. 9 Optimale Strategie

Wir werden diese Frage nach einer optimalen Strategie durch Simulation zu beantworten versuchen. Läßt man den Rechner für verschiedene Wurfzahlen je Runde eine Weile spielen, so geben die Resultate sicher einen Hinweis auf eine optimale Strategie. Das Verfahren ist zeitraubend und wir suchen nach einem besseren Weg.

Der Grundgedanke zur Simulation läßt sich folgendermaßen beschreiben: Es sitzen N Spieler in einer Runde zusammen und s spielen „Die verflixte Eins". Der erste Spieler S1 ist sehr vorsichtig und würfelt stets nur einmal. Dann hört er auf und gibt an seinen Nachbarn ab. Sein Nachbar S2 würfelt stets zweimal und hört dann auf. Spieler S3 hört nach dem dritten Wurf auf und so weiter. Spieler S_i würfelt also stets i-mal und hört dann auf, um an seinen Nachfolger weiterzugeben. Welcher Spieler hat auf Dauer die besten Erfolgsaussichten?

Die computergerechte Aufbereitung des Simulationsverfahrens ist im Struktogramm **Fig. 9** erfolgt.

Struktogramme haben gegenüber **Programmablaufplänen** zwei wesentliche Vorzüge:

1. Struktogramme führen zu übersichtlichen Programmen, da sie nicht zu unkontrollierten Sprüngen verführen.
2. Es gibt mehr Platz für Kommentare. Dadurch sind Struktogramme leichter lesbar als Programmablaufpläne, die meist ohne erläuternden Text nicht auskommen.

Allerdings sind Struktogramme nicht so nah an der **Maschinensprache** des PTR orientiert, wie Programmablaufpläne. Da außerdem die das Auge führenden Linien fehlen, sind Struktogramme für den Anfänger nicht so unmittelbar verständlich wie Programmablaufpläne. Schüler geben daher zunächst den Programmablaufplänen den Vorzug. Man sollte daher im Unterricht nach Einführung der Programmablaufpläne eine Zeitlang beide Formen nebeneinander benutzen.

Will man zu einem Struktogramm das zugehörige Rechnerprogramm schreiben, so hält man sich zweckmäßigerweise an folgende *Regeln für das Codieren von Struktogrammen:*

1. Speichertabelle anlegen. Diese ordnet jeder Variablen (soweit nötig) ein Register zu.
2. Label festlegen und in das Struktogramm eintragen.
   2.1 Eingabelabel und Unterprogrammlabel festlegen.

Programmablaufplan: Graphische Darstellung des logischen Programmablaufs mit nach DIN 66001 genormten Sinnbildern, wobei direkt die Ablauffolge (der *Programmfluß*) sichtbar gemacht wird. Im „Jahrbuch '80" ist diese Methode von *W. Schneider* beschrieben („Graphische Darstellung von Programmablaufplänen").

Maschinensprache: Datenverarbeitende Maschinen arbeiten *binär* (zweiwertig), d.h. sie leiten alle Abläufe und Reaktionen aus dem Vorhandensein oder Fehlen elektrischer Größen ab (z.B. Spannung 5 V oder 0 V). Aber nicht nur die definierten Kombinationen von solchen „zweiwertigen" Signalen nennt man Maschinensprache, sondern auch die leichter lesbaren *hexadezimalen Darstellungen* der binären Schaltanweisungen.

Var	*N*	*K*	*i*	*k*	*R*	*s*	*z*
Reg	59	58	57	56	55	54	00

Fig. 10 Speichertabelle

2.2 Alternativen:
Ja-Zweig und nachfolgenden Strukturblock mit Labeln versehen. (Empfehlung: zuerst Nein-Zweig codieren, mit GOTO abschließen, dann Ja-Zweig codieren).

2.3 Schleifen:
Oberkante und Unterkante mit Label versehen. (Ausnahme: Repeat-until-Schleife = Schleife mit Abbruchbedingung am Schluß: Duale Abfrage erspart Label).

3. Codierung zeilenweise. Label unterstreichen. Dadurch behält man während der Arbeit des Codierens besser den Überblick und etwa notwendiges Redigieren von Programmen wird erleichtert.

Die Speichertabelle ist gemäß **Fig. 10** angelegt und die verwendeten Label sind in **Fig. 11** aufgelistet. Auf eine Eintragung in das Struktogramm gemäß Codierungsregel 2 wurde verzichtet, um das Druckbild nicht unübersichtlich werden zu lassen. Die Dokumentation sollte man nicht wie das Programmieren nach Regel 3 zeilenweise, sondern möglichst durch ein Druckerprotokoll ausführen. Allerdings empfiehlt es sich, wie auch hier geschehen, die einzelnen Programmsegmente entsprechend den Strukturblöcken zu unterteilen. Das nach den Codierungsregeln entstandene Programm ist in **Fig. 12** aufgelistet.

Will man die Simulation mit $s = 0{,}2$ als Startzahl, $N = 10$ Spielern und $K = 200$ Runden durchführen, so wird das Programm durch die Tastenfolge 0.2 A 10 B 200 C D gestartet. Nach ca. 35 Minuten Rechenzeit erhält man die Simulationsergebnisse (**Fig. 13**). Der Verfasser muß einräumen, daß die Startzahl so gewählt wurde, daß die Simulationsergebnisse den zu erwartenden Werten entsprechen. Im Unterricht wird jeder Schüler, möglichst als Hausaufgabe, die Simulation mit einer anderen Startzahl durchführen. Die theoretische Diskussion des Problems führt dann darauf, daß zwei optimale Wurfzahlen existieren.

```
001  11  A
008  12  B
014  13  C
020  14  D
026  22  INV
034  23  LNX
077  24  CE
```

Fig. 11 Labelliste

ADR	Code	Taste	ADR	Code	Taste	ADR	Code	Taste
000	76	LBL	031	42	STO	062	43	RCL
001	11	A	032	57	57	063	55	55
002	99	PRT	033	76	LBL	064	74	SM*
003	47	CMS	034	23	LNX	065	57	57
004	42	STO	035	43	RCL	066	01	1
005	54	54	036	54	54	067	44	SUM
006	91	R/S	037	85	+	068	57	57
007	76	LBL	038	89	π	069	43	RCL
008	12	B	039	95	=	070	57	57
009	99	PRT	040	45	YX	071	32	X:T
010	42	STO	041	05	5	072	43	RCL
011	59	59	042	95	=	073	59	59
012	91	R/S	043	22	INV	074	77	GE
013	76	LBL	044	59	INT	075	23	LNX
014	13	C	045	42	STO	076	76	LBL
015	99	PRT	046	54	54	077	24	CE
016	42	STO	047	65	×	078	01	1
017	58	58	048	06	6	079	44	SUM
018	91	R/S	049	85	+	080	56	56
019	76	LBL	050	01	1	081	43	RCL
020	14	D	051	95	=	082	56	56
021	98	ADV	052	59	INT	083	32	X:T
022	01	1	053	69	DP	084	43	RCL
023	42	STO	054	20	20	085	58	58
024	56	56	055	32	X:T	086	22	INV
025	76	LBL	056	01	1	087	67	EQ
026	22	INV	057	67	EQ	088	22	INV
027	00	0	058	24	CE	089	00	0
028	42	STO	059	32	X:T	090	22	INV
029	55	55	060	44	SUM	091	90	LST
030	01	1	061	55	55	092	91	R/S

Fig. 12 Simulation eines Würfelspiels. Entwicklung einer optimalen Strategie

```
  0.2    s        1636.   03
 10.     N        1875.   04
200.     K        1997.   05
                  1963.   06
1118.    00       1903.   07
 700.    01       1686.   08
1280.    02       1321.   09
                  1337.   10
```

Fig. 13 Simulationsergebnisse