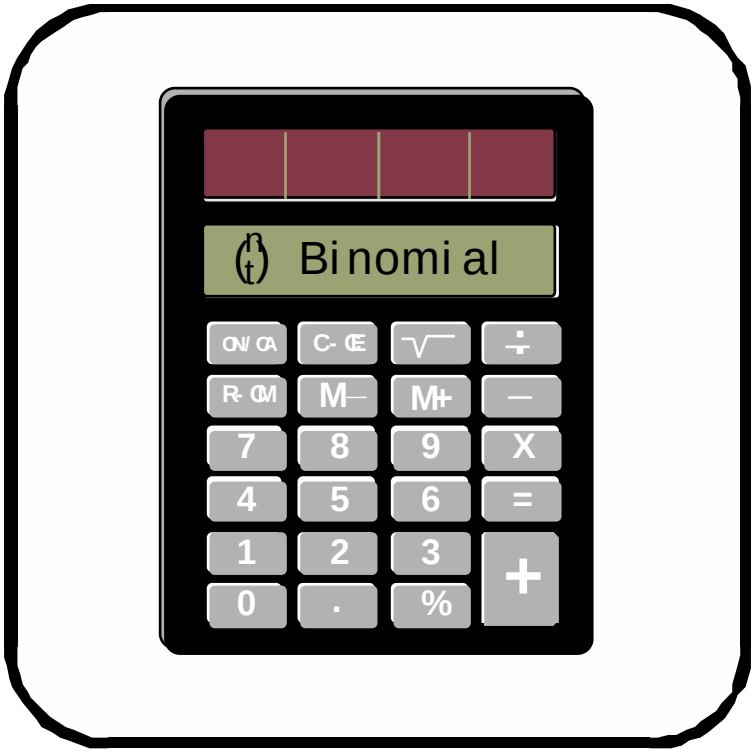
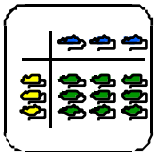
Binomialkoeffizienten





Werkstatt Multiplikation
Posten: **Binomialkoeffizienten**

Informationsblatt für die Lehrkraft

Thema: Binomialkoeffizienten auf dem Taschenrechner

Schultyp: Mittelschule, technische Berufsschule, Fachhochschule

Vorkenntnisse: Grundlagen der Programmierung: ARRAY, INTEGER, REAL, FOR, INT(), einfache I/O-Operationen auf Tastatur/Bildschirm.
Keine Kenntnis von Unterprogrammen oder Prozeduren nötig.
Keine speziellen mathematischen Vorkenntnisse nötig.

Bearbeitungsdauer: 120 Minuten

Fassung vom: 15.9.95

Schulerprobung: nein

Übersicht

Der Posten besteht aus dem Auftrag ein Programm zur **Berechnung der Binomialkoeffizienten** zu erstellen. Das Programm kann auf einem **Taschenrechner** oder auf einem **Computer1** erstellt werden.

Ein **Rahmenprogramm wird vorgegeben**. Dadurch reduziert sich der Programmieraufwand auf die **Multiplikation** einer grossen Zahl2 mit einer zweistelligen Zahl und die **Division** einer grossen Zahl durch eine zweistellige Zahl. Das Rahmenprogramm liegt als Pseudo-Code vor. Es muss vor Abgabe des Postens an die Klasse in eine **konkrete Programmiersprache** umgesetzt werden.

Der Postenabsolvent muss **kein Spezialist** sein. Der Posten ist so konstruiert, dass er von **Programmieranfängern** erfolgreich gemeistert werden kann.

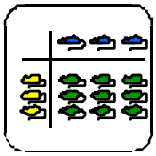
Motivierend ist, dass das erstellte Programm die Fähigkeiten des eigenen Taschenrechners erweitert.

Der Begriff **Pascalsches Dreieck** wird im Pseudo-Code und in der Zusammenfassung verwendet, aber nicht erklärt.

---

1 Im Rest des Textes schreibe ich immer nur vom Taschenrechner. Für den Computer gilt natürlich alles sinngemäss.

2 Im ganzen Text verstehe ich als grosse Zahl eine ganze Zahl, die mehr gültige Stellen hat als der Taschenrechner. Ein üblicher Taschenrechner hat 11-12 gültige Stellen. 15! hat 13 Stellen.



Lernziele

- Der Postenabsolvent bekommt ein Beispiel, wie er dem Taschenrechner den **Umgang mit grossen Zahlen** beibringen kann. Wenn er später wieder einmal mit grossen Zahlen rechnen wird, erinnert er sich an die hier verwendete Methode und Datenstruktur und passt sie an das neue Problem an.

- Der Postenabsolvent beschäftigt sich mit den Binomialkoeffizienten. Er kennt die Definition der Binomialkoeffizienten nun auswendig.

- Der Postenabsolvent erkennt, dass oft mehrere Möglichkeiten bestehen, wie ein Programm etwas berechnen kann. Wenn er weitere Programme schreibt, wird er sich die Zeit nehmen die **verschiedenen Möglichkeiten** gegeneinander **abzuwägen**.

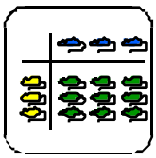
Material

- *Theorie*: Binomialkoeffizienten auf dem Taschenrechner (in diesen Unterlagen enthalten)
- *Material:* **Programmierbarer Taschenrechner** mit Handbuch[3]. Die Aufgaben können aber auch auf einem **Computer** gelöst werden.

Quellen

Fachred. d. Bibliograph. Ins.: *Duden "Die Mathematik"* Mannheim; Wien; Zürich (Bibliographisches Institut, 1982)

3 Um die Befehle und deren Syntax nachschlagen zu können.



Hinweise, Lösungen

Lösung Auftrag 2

Aufgabenstellung:

Vervollständige das Programm zur Berechnung der Binomialkoeffizienten. Dazu schreibst Du **zwei Programmteile:**

1. Die **Multiplikation** einer grossen Zahl mit einer zweistelligen Zahl
2. Die **Division** einer grossen Zahl durch eine zweistellige Zahl

Diese zwei Programmteile **integrierst** Du dann in das vorgegebene Rahmenprogramm. Damit entsteht ein Programm zur Berechnung der Binomialkoeffizienten. Zugelassen werden nur **maximal zweistellige Argumente** (Zahlen von 0 bis 99).

Lösung:

Die Multiplikation:

Für jedes Arrayelement von `GrosseZahl` muss

- das Arrayelement mit dem zweistelligen **Faktor multipliziert** werden
- ein allfälliger **übertrag** von der vorhergehenden Stelle **addiert** werden
- das Arrayelement **normalisiert** werden, d.h. auf 9 (Dezimal-) Stellen reduziert werden und der übertrag für die nächste Stelle gebildet werden.

In Pseudo-Code formuliert:

```
{ ----- Die Multiplikation GrosseZahl:= GrosseZahl * Faktor }
{ Initialisierung }
Uebertrag:= 0;
{ Jedes Arrayelement multiplizieren }
FOR j:= 1 TO 4 DO
    Tmp:= GrosseZahl[j] * Faktor + Uebertrag;
    { ----- Zahl normalisieren und Uebertrag bilden }
    { Die 10. und 11. Stelle gehören zum nächsten Arrayelement }
    Uebertrag:= INT(Tmp / 1000000000);
    { Die untersten 9 Stellen bleiben beim aktuellen Arrayelement }
    GrosseZahl[j]:= Tmp - Uebertrag * 1000000000;
END;
```

Die Division:

Für jedes Arrayelement von `GrosseZahl` muss

- ein allfälliger **übertrag** von der vorhergehenden Stelle **addiert** werden
- das Arrayelement durch den zweistelligen Divisor **dividiert** werden

In Pseudo-Code formuliert:

```
{ ----- Die Division GrosseZahl:= GrosseZahl / Divisor }
{ Initialisierung }
Uebertrag:= 0;
{ Jedes Arrayelement dividieren }
FOR j:= 4 DOWNTO 1 DO
    { übertrag addieren }
    Tmp:= GrosseZahl[j] + Uebertrag * 1000000000;
    { Dividieren und Uebertrag (=Rest) ermitteln }
    GrosseZahl[j]:= INT(Tmp / Divisor);
    Uebertrag:= Tmp - Divisor * GrosseZahl[j];
END;
```

Die zusätzlich verwendeten Variablen müssen im Rahmenprogramm noch deklariert werden:

```
Tmp:        REAL;                   { Zwischenresultat Multiplikation }
Uebertrag:  REAL;                   { Uebertrag Multiplikation }
j:          INTEGER;                { Laufvariable }
```

Einige Resultate:

$\binom{32}{27} = \binom{32}{5} = 201'376$

$\binom{45}{6} = 8'145'060$

$\binom{22}{12} = 646'646$

$\binom{49}{6} = 13'983'816$

$\binom{99}{19} = 10'719'667'408'076'193'6594$

Lehrer-Lernkontrolle / Test

Aufgabe 1 (schriftlich)

Du hast das Binomialprogramm vervollständigt. Die grosse Zahl ist in einem Array von REAL gespeichert:

```
GrosseZahl: ARRAY [1..4] OF REAL;
```

Nun liest Du in der Bedienungsanleitung Deines Taschenrechners, dass dieser noch den **Datentyp MASCH** kennt. Mit diesem Datentyp rechnet der Taschenrechner **viel schneller** als mit REAL. Im Typ MASCH können ganze Zahlen von 0 bis 264-1 (1.844671019) dargestellt werden. Du willst nun das Programm beschleunigen, indem Du statt des Datentyps REAL, den Datentyp MASCH verwendest.

Wie ändert sich die Deklaration (VAR ...) der Variablen GrosseZahl? überdenke insbesondere auch die Arraygrösse!

Aufgabe 2 (schriftlich)

Das Binomialprogramm aus Auftrag 2 lässt nur **zweistellige Argumente** zu. Wie musst Du das Programm verändern um **vierstellige Argumente** zuzulassen?

Ich **erwarte zwei** wichtige Punkte, die zu ändern sind. Du musst nicht das Programm neu schreiben. Es genügen die Hinweise in Worten, was zu ändern ist.

Lösungen und Taxierung

Aufgabe 1 (schriftlich)

Lösung: In einer Variablen vom Datentyp MASCH können 19 (Dezimal-) Stellen gespeichert werden. Damit bei einer Multiplikation mit einer zweistelligen Zahl das Resultat noch im Zahlenbereich von MASCH bleibt, sollten 17 (Dezimal-) Stellen pro Variable gespeichert werden. Der grösste Binomialkoeffizient bei zweistelligen Argumenten umfasst 30 (Dezimal-) Stellen. (Diese Information ist auf dem Auftragsblatt enthalten). Dadurch kann das Array **GrosseZahl auf 2 Elemente reduziert** werden:

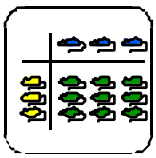
```
GrosseZahl: ARRAY [1..2] OF MASCH;
```

Taxierung: Im Auftrag dieses Postens ist die Datenstruktur für die grosse Zahl vorgegeben. Es wird jedoch nicht erklärt, wie die Anzahl Stellen pro Arrayelement ermittelt werden. Da die Lösung aber auf der Hand liegt, taxiere ich die Aufgabe als höchstens **K3**.

Aufgabe 2 (schriftlich)

Lösung: Bei der Multiplikation mit der vierstelligen Zahl darf kein überlauf auftreten. Deshalb dürfen **pro Arrayelement** nur **7 (Dezimal-) Stellen** gespeichert werden. Ich bin dabei wieder von 11 signifikanten Stellen pro REAL-Zahl ausgegangen, wie auch im Auftrag 2.

Taxierung: Die Bedingungen wurden geändert. Die Änderung geht über das reine Einsetzen von anderen Zahlen hinaus, da die Datenstruktur angepasst werden muss. Die Aufgabe ist deshalb **K3**.



Werkstatt Multiplikation Auftragsblatt
Posten: **Binomialkoeffizienten**

Was soll ich hier tun?

Wieviele Möglichkeiten hast Du einen **Lottoschein** auszufüllen? Die Mathematiker haben sich darum gekümmert. Die Lösung ist einfach: Um in einem Feld mit 45 Zahlen 6 Zahlen anzukreuzen, gibt es $\binom{45}{6}$ Möglichkeiten[4].

Leider können viele Taschenrechner das Resultat nicht direkt ausrechnen. Ein Grund dafür ist, dass sie diese **Funktion** (zur Berechnung der Binomialkoeffizienten) **nicht kennen**. Aber es gibt noch ein zweites Problem: Die auftretenden **Zwischenresultate** können sehr gross werden.

An diesem Posten kannst Du Deinem programmierbaren Taschenrechner diese Funktionen beibringen.

Der Posten besteht aus den **folgenden zwei Aufträgen**:

(1) Studiere die Theorie "*Binomialkoeffizienten auf dem Taschenrechner*"
(20 - 30 Minuten)

(2) Vervollständige das Programm zur Berechnung der Binomialkoeffizienten. Dazu schreibst Du **zwei Programmteile**:

1. Die **Multiplikation** einer grossen Zahl mit einer zweistelligen Zahl
2. Die **Division** einer grossen Zahl durch eine zweistellige Zahl

Diese zwei Programmteile **integrierst** Du dann in das vorgegebene Rahmenprogramm. Damit entsteht ein Programm zur Berechnung der Binomialkoeffizienten. Zugelassen werden nur **maximal zweistellige Argumente** (Zahlen von 0 bis 99).

Das **Rahmenprogramm** findest Du am Ende dieses Kapitels.

Die **Datenstruktur für die grosse Zahl** ist ein Array `GrosseZahl[1..4]`. In jedem Arrayelement sind 9 (Dezimal-) Ziffern der Zahl abgespeichert. In `GrosseZahl[1]` sind die niederwertigsten 9 Stellen.

Ein Beispiel:
```
GrosseZahl[1]=     0
GrosseZahl[2]=   666’777’888
GrosseZahl[3]=   333’444’555
GrosseZahl[4]=       111’222
```
steht für die Zahl 111’222’333’444’555’666’777’888’000’000’000

Mit dieser Datenstruktur kannst Du Zahlen mit bis zu 4*9= 36 Stellen darstellen. Der **grösste Wert**, den die Binomialkoeffizienten für zweistellige Argumente annehmen kann, hat **30 Stellen**.

Vor Deinem Programmteil gilt jeweils:

· Die grosse Zahl ist in `GrosseZahl[1..4]` gespeichert.

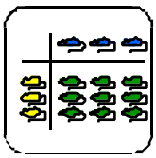
· Der Multiplikator ist in `Faktor` gespeichert, bzw. der Divisor ist in `Divisor` gespeichert.

Nach Deinem Programmteil muss gelten:

· In `GrosseZahl[1..4]` ist das Resultat der Multiplikation bzw. Division.

---

4 sprich "45 tief 6"



Werkstatt Multiplikation Auftragsblatt
Posten: **Binomialkoeffizienten**

Füge im Rahmenprogramm an der vorgegebenen Stelle Dein Multiplikations- und Divisionsprogramm ein.

Ein kleiner **Tip**: Wahrscheinlich brauchst Du die INT-Funktion. Die INT-Funktion liefert den ganzzahligen Teil des Arguments zurück. Also z.B. INT(3.4)= 3, INT(5.9)= 5. Manchmal heisst die Funktion auch TRUNC oder IP.

Beachte bitte:

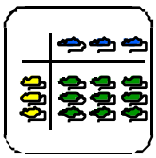
- Pro Arrayelement in `GrosseZahl` sind genau 9 (Dezimal-) Stellen gespeichert.
- Vom Rahmenprogramm darfst Du die Variablen `GrosseZahl` und `i` verändern. Wenn Du weitere Variablen benötigst, musst Du sie im Rahmenprogramm definieren.
- Du kannst davon ausgehen, dass bei der **Multiplikation kein überlauf** (Overflow) und bei der **Division kein Rest** auftritt. Damit kannst Du mit der eingebauten Genauigkeit des Taschenrechners arbeiten.

Ich erwarte das Programm sauber dokumentiert auf Papier und auf Deinem Taschenrechner implementiert. Ich werde Deine Lösung mit Dir besprechen. Dabei ist mir **wichtig**:

- Ist das Programm so dokumentiert, dass eine Drittperson die Lösung einfach nachvollziehen kann?
- Funktionieren die Programmteile Multiplikation und Division korrekt?
- Sind die Programmteile korrekt eingebunden, d.h. werden nur diejenigen globalen Variablen verändert, bei denen dies erlaubt ist?

Wenn Du diese drei Punkte einhälst, ist der **Auftrag erfüllt**. Ich werde einige Binomialkoeffizienten mit Deinem Taschenrechner berechnen.

Diese beiden Aufträge sollte **jeder für sich** bearbeiten.



Das Rahmenprogramm in Pseudo-Code

```
{ Programm zur Berechnung der Binomialkoeffizienten }

{ ----- Deklaration der Variablen }
VAR i:          INTEGER;                { Laufvariable }
    GrosseZahl: ARRAY [1..4] OF REAL;   { Resultat der Funktion }
    n, k:       INTEGER;                { Argumente der Funktion }
    Faktor:     INTEGER;                { Aktueller Multiplikator }
    Divisor:    INTEGER;                { Aktueller Divisor }

{ ----- Initialisierung: GrosseZahl:= 1 }
GrosseZahl[1]:= 1;
FOR i:= 2 TO 4 DO
    GrosseZahl[i]:= 0
END;

{ ----- Funktionsargumente vom Benutzer eingeben lassen }
WriteLn "Binomialkoeffizienten: n= ";
INPUT n;
WriteLn "Binomialkoeffizienten: k= ";
INPUT k;

{ Für die Argumente muss gelten: n>=k und k>=0 und n<=99 }
IF n > 99 THEN HALT END;
IF k <  0 THEN HALT END;
IF n <  k THEN HALT END;

{ ----- Berechnung des Binomialkoeffizient (n tief k) }
{ ----- Möglichst wenige Schleifendurchläufe mit der Identität }
{         (n tief k) = (n tief n-k) }
IF 2*k > n THEN k:= n-k END;

{ ----- Im Pascalschen Dreieck der Zeile nachgehen }
FOR i:= 0 TO k-1 DO
    { ----- (n tief k+1) = (n tief k) * ( (n-k)/(k+1) ) }
    Faktor:= n-i;
    Divisor:= i+1;
    { ----- Die Multiplikation GrosseZahl:= GrosseZahl * Faktor }

    Hier kommt Dein Multiplikations-Code

    { ----- Die Division GrosseZahl:= GrosseZahl / Divisor }

    Hier kommt Dein Divisions-Code

END; { FOR i }

{ ----- Ausgabe des Resultats }
{       Es werden einfach alle Arrayelemente von GrosseZahl ausgegeben. }
{       Zuerst das hochwertigste Element }
WriteLn n, " tief ", k, " = ";
FOR i:= 4 DOWNTO 1 DO
    WriteLn GrosseZahl[i];
END;
```

Binomialkoeffizienten auf dem Taschenrechner

Wieviele Möglichkeiten hast Du einen **Lottoschein** auszufüllen? Die Mathematiker wissen es: Um von 45 Zahlen 6 Zahlen anzukreuzen, gibt es $\binom{45}{6}$ Möglichkeiten (sprich "45 tief 6").

Die **Definition** von $\binom{n}{t}$ ist: $\binom{n}{t} = \frac{n!}{(n-t)!\ t!}$.

Dabei ist ! der **Fakultätsoperator**. n! ist definiert als n!= n(n-1)(n-2)...21. Also 5!= 54321= 120. Ein Spezialfall ist: 0!= 1.

Um $\binom{100}{5}$ zu berechnen, rechnest Du: $\frac{100!}{(100-5)!\ 5!} = \frac{100!}{95!\ 5!}$.

Eine **Möglichkeit** diesen Bruch zu berechnen, ist einfach drauflos zu rechnen[5]:

1. 100! liefert 9.3326E157
2. 95!*5! liefert 1.2396E150
3. Die Division des Resultats 1. durch das Resultat 2. liefert 75'287'520

Leider hat diese Möglichkeit einige **Nachteile**:

1. Obwohl das Resultat am Schluss klein ist, sind die **Zwischenresultate sehr gross**, eventuell sogar zu gross für Deinen Taschenrechner. Der Taschenrechner wird dann die Berechnung mit einem überlauf (Overflow-Error) ablehnen.
2. Wieviele Multiplikationen braucht die Berechnung?
   100! = 99 Multiplikationen
   95!*5!= 94+4+1= 99 Multiplikationen
   Im **Total** also **198 Multiplikationen!** Diese Methode den Bruch zu berechnen ist **sehr aufwendig!**

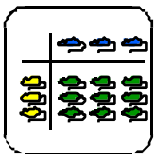
Es gibt eine **bessere Lösung**: Du kannst den Aufwand verringern, indem Du den Bruch vorher **kürzst**:

$$\frac{100!}{95!\ 5!} = \frac{100\cdot 99\ 98\ 97\ 96\ 95!}{95!\ 5!} = \frac{100\cdot 99\ 98\ 97\ 96}{5!} = \frac{90'345'024'00}{120} = 75'287'520$$

Das sind nur noch 4+4= **8 Multiplikationen** und das grösste Zwischenresultat hat nur noch 10 (Dezimal-) Stellen, statt 158 Stellen.

---

5 Einfach drauflos zu rechnen hat sogar einen Namen. Es wird *Brute Force-Methode* genannt. Der Name kommt daher, dass mit *'brutaler Kraft'* einfach drauflos gerechnet wird. Meist ist es sinnvoll etwas Zeit in das Suchen von besseren Lösungswegen zu investieren.



Werkstatt Multiplikation Theorie
Posten: **Binomialkoeffizienten**

Für die Berechnung mit dem Taschenrechner ist das **Kürzen** aber **ungeeignet**. Es gibt aber zwei Formeln, die für die Taschenrechnerversion nützlich sind:

1. $\binom{n}{0} = 1$

2. $\binom{n}{t+1} = \binom{n}{t} * \frac{n-t}{t+1}$

Du beginnst mit t= 0 und berechnest den Binomialkoeffizienten $\binom{n}{0} = 1$. Dann berechnest Du mit der zweiten Formel den Binomialkoeffizienten für t= 1, t= 2, usw. bis t den gewünschten Wert hat.

Ein kleines *Beispiel*: $\binom{6}{3} = ?$

$$\binom{6}{0} = 1, \quad \binom{6}{1} = 1 * \frac{6-0}{0+1} = 6, \quad \binom{6}{2} = 6 * \frac{6-1}{1+1} = 15, \quad \binom{6}{3} = 15 * \frac{6-2}{2+1} = 20$$

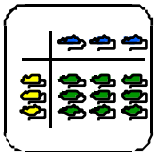
Nach dieser Rechenvorschrift berechnet das Programm an diesem Posten die Binomialkoeffizienten. Es nutzt aber noch eine zusätzliche Formel aus: $\binom{n}{t} = \binom{n}{n-t}$, z.B. $\binom{100}{95} = \binom{100}{5}$.

Wozu ist diese Formel gut? Berechne einmal $\binom{45}{43}$ nach der obigen Vorschrift! Dazu musst du die Formel 2. 43 mal anwenden!

Da ist doch $\binom{45}{2} = 1 * \frac{45-0}{0+1} * \frac{45-1}{1+1} = 45 * 22 = 990$ **einfacher und schneller**, oder?

Die Binomialkoeffizienten sind immer nur **ganze Zahlen**. Es kann also nicht geschehen, dass bei der Division ein Rest auftritt. Zumindest nicht, wenn Du vorher mit dem Zähler des Bruches multipliziert hast!

Schau Dir nun die **Zusammenfassung** an. Berechne dann $\binom{10}{3}$ mit allen drei Methoden. Dies hilft Dir, die Unterschiede zwischen den Methoden zu verstehen.



Werkstatt Multiplikation Theorie
Posten: **Binomialkoeffizienten**

Zusammenfassung: Binomialkoeffizienten auf dem Taschenrechner

Du hast drei Methoden zur Berechnung der Binomialkoeffizienten kennengelernt:

1. Die Brute-Force-Methode:

Direkte Berechnung mit der **Definition** der Binomialkoeffizienten:

$$\binom{n}{t} = \frac{n!}{(n-t)!\ t!} \text{ mit } n \geq t \geq 0$$

2. Kürzen

Vor dem Ausrechnen **kürzen**. Diese Methode ist besonders geeignet für die **Berechnung von Hand**:

$$\frac{100!}{95!\ 5!} = \frac{100 \cdot 99 \cdot 98 \cdot 97 \cdot 96 \cdot 95!}{95!\ 5!} = \frac{100 \cdot 99 \cdot 98 \cdot 97 \cdot 96}{5!} = \frac{9'034'502'400}{120} = 75'287'520$$

3. Pascalsches Dreieck

Hier werden zwei Formeln ausgenutzt:

1. $\binom{n}{0} = 1$
2. $\binom{n}{t+1} = \binom{n}{t} * \frac{n-t}{t+1}$

Zusätzlich kann duch die Formel $\binom{n}{t} = \binom{n}{n-t}$ der Rechenaufwand bei 2t > n reduziert werden.