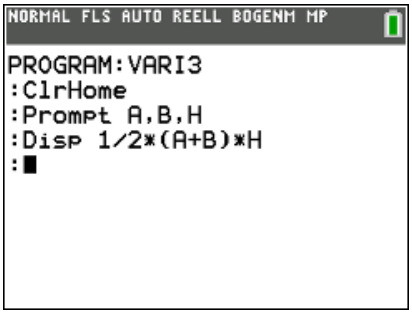
Lektion 2: Variablen und Terme	Übung 3: Ausdrücke erstellen, berechnen und in Variablen abspeichern
In der dritten Übung von Lektion 2 geht es um die Verwendung von Ausdrücken (Termen) und das Abspeichern von Werten in Variablen innerhalb eines Programmes.	**Lernziele:** • Programmierung mathematischer Ausdrücke. • Rangfolge der mathematischen Operatoren. • Unterscheidung zwischen mathematischen und Cumputervariablen. • Berechnung von Ausdrücken (Termen). • **Abspeichern (Store)** der Ergebnisse von Termen in Variablen.

Ausdrücke (Terme)

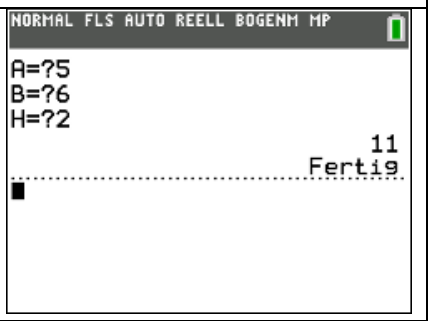
Zeichenkombinationen wie **A^2** und **A+B** werden Terme (Ausdrücke) genannt. Terme findet man in mathematischen Formeln; z.B. lautet die **Formel** für die Dreiecksfläche A: A = ½·A·B . Darin ist ½·A·B der **Term**.

Das Programm berechnet den Term unter Verwendung der aktuellen Werte der Variablen, so dass man als Ergebnis einen numerischen Wert erhält. Dabei werden die Rechenregeln der Algebra verwendet. (Punkt- vor Strichrechnung, Klammerregeln).

Beispiel: Das abgebildete kleine Programm berechnet die Fläche eines **Trapezes** mit den Seiten **A** und **B** und der Höhe **H**.



Bei den Taschenrechnern der TI 84-Familie sind nur Variablennamen erlaubt, die aus *einem* Buchstaben bestehen. Die Variablen **B1** und **B2** würde der Taschenrechner als **B·1** und **B·2** interpretieren und bei falscher Verwendung eine Fehlermeldung erzeugen. Ebenso würde es der Variablen **AB** ergehen, die als **A·B** interpretiert würde. Dabei handelt es sich um sogenannte *implizierte* Multiplikationen, da das Multiplikationszeichen zwischen den einzelnen Zeichen *intern* als *vorhanden* angenommen wird.



Mathematische Ausdrücke und Ausdrücke in Programmen

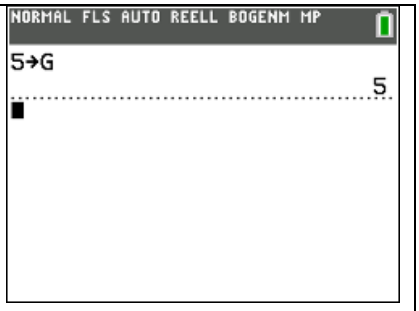
Obwohl es zwischen beiden viele Gemeinsamkeiten gibt, so existieren doch wichtige Unterschiede. Der Wichtigste ist wohl, dass man in der Mathematik mit den Symbolen rechnet, die ja ganz allgemein Stellvertreter für Zahlen sind, und erst bei Bedarf Zahlen einsetzt. Beim Computer hingegen wird stets mit Zahlen gerechnet; die Variablennamen sind also lediglich Bezeichnungen für Zahlen.

In der Mathematik werden Formeln eingesetzt, um zwischen verschiedenen Dingen wie z.B. Längen und Fläche eine Beziehung herzustellen. In Programmen hingegen werden nur Terme eingesetzt, die dann eine Zahl als Ergebnis liefern. Schreibt man das Programm, so schreibt man den Term, aber läuft das Programm, wird der Term berechnet und erzeugt ein Ergebnis, das dann weiterverwendet werden kann.

Hinweis: Für Anfänger sehr verwirrend ist die Programmzeile **x+1→x**. Sie bedeutet in der Syntax des TI Basic, dass **1** zu **x** addiert und das Ergebnis wieder in **x** abgespeichert wird. Das **x** auf der linken Seite hat also einen anderen Wert als das auf der rechten Seite. Die Befehlszeile ist als *Zähler* bekannt, denn bei jedem Durchlauf z.B. in einer Schleife wird **x** um **1** erhöht. In anderen Sprachen wie z.B. B.A.S.I.C und Lua wird dieser Befehl in der Form x = x + 1 geschrieben, was mathematisch sicher falsch ist, aber innerhalb eines Programmes mit der besonderen Syntax richtig.

Werte in Variablen abspeichern mit dem Zuweisungs-Operator

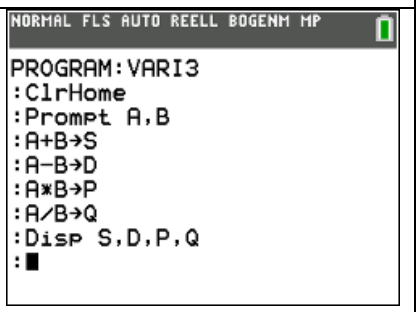
Der Operator STO▶ wird verwendet, um das Ergebnis eines Terms in einer Variablen abzuspeichern (der Variablen zuzuweisen).
Drückt man die Taste STO▶, so wird stets das Zeichen → dargestellt.
Drückt man ENTER, so sieht man auf dem HOME-Bildschirm das Ergebnis des Terms und die Variable (im Beispiel **G**) hat nun den Wert **5**. Vor dem Pfeil muss stets eine **Zahl** oder ein **Term** stehen, der einen Wert erzeugt. Dieser Wert wird dann der **Variablen** zugewiesen. Das Zeichen hinter dem Pfeil muss deshalb stets der Name einer **Variablen** sein.



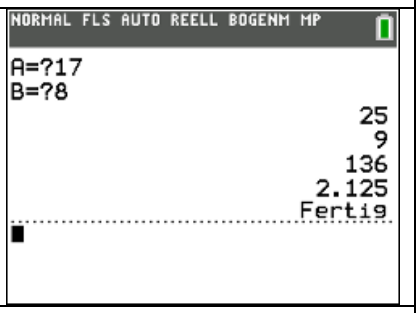
Programmieren mit Wertezuweisungen

Es soll ein Programm geschrieben werden, bei dem zwei Zahlen eingegeben werden sollen, aus denen dann die Summe, die Differenz, das Produkt und der Quotient berechnet und abgespeichert werden. Die Ergebnisse sollen dann ausgegeben werden.

1. Ein neues Programm wird gestartet**.**
2. Mit **Prompt** werden die zwei Variablen **A** und **B** eingelesen.
3. Die Terme sollen in vier anderen Variablen **S**, **D**, **P** und **Q** gespeichert werden.
4. **S**, **D**, **P** und **Q** werden dann mit **Disp** ausgegeben.



Das Programm kann nun gestartet werden. Gibt man für **A** und **B** Zahlen ein, so sollte man sie so wählen, dass man die Resultate leicht überprüfen kann. So etwas nennt man dann einen *Testdurchlauf*.



Hinweis: Der Zuweisungsbefehl in TI Basic ist in seiner Form einzigartig, da zuerst der Term kommt, dann der "store"-Operator und dann die Variable. Dadurch ist die Zeile leicht lesbar. In den meisten anderen Programmiersprachen ist die Reihenfolge genau umgekehrt wie z.B. in S = A + B. Hier muss man rückwärts lesen, denn der Computer berechnet erst den Ausdruck auf der rechten Seite und weist dann den Wert der Variablen auf der linken Seite zu.

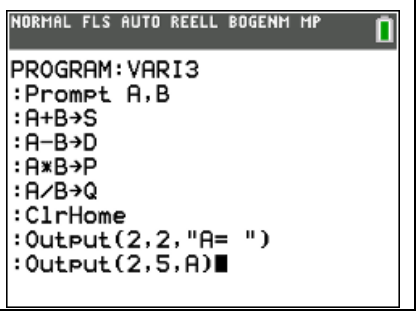
Hinweis: Die Verwendung von Variablen hat viele Vorteile. Mit Variablen lässt sich ein Ergebnis einfacher darstellen. Außerdem können dieselben Variablen in verschiedenen Berechnungen verwendet werden. Wird dieselbe Zahl an verschiedenen Stellen des Programmes verwendet, so sollte man sie in einer Variablen abspeichern, so dass man nur noch den Namen der Zahl im Programm verwenden muss.

Übung: Verbesserungen

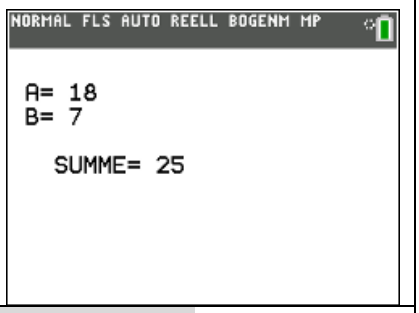
Mit **Output(** anstelle von **Disp** kann die Darstellung der Ergebnisse verbessert werden, denn man kann sowohl die Eingaben als auch die Ergebnisse besser und übersichtlicher darstellen.

Rechts ist ein Teil des Programmes abgebildet sowie eine mögliche Darstellung auf dem Display. Das Programm selbst ist deutlich länger …

Der **Pause**-Befehl sollte nicht vergessen werden, um die Meldung **Fertig** zu



unterdrücken. Und man sollte an die unterschiedlichen Bildschirmauflösungen der Taschenrechnermodelle denken (**TI-84 Plus CE-T**: 24 Spalten / 10 Zeilen; **TI-84 Plus**: 16 Spalten / 8 Zeilen).



Hinweis: Man sollte die Schüler auffordern, auch andere mathematische Formeln zu programmieren und mit den Ausgabemöglichkeiten zu experimentieren. In der Anwendung zu dieser Lektion findet sich ein Programmierbeispiel mit drei Formeln.