

# Table of Contents

|  |   |
|--|---|
| <b>Cache-Variablen</b> .....                   | 2 |
| <b>Einleitung</b> .....                        | 2 |
| <b>Kontrollbereich</b> .....                   | 3 |
| <b>Variablenbereich</b> .....                  | 3 |
| <b>Formelsyntax</b> .....                      | 4 |
| Typen von Variablen .....                      | 5 |
| Numerische Operatoren .....                    | 5 |
| Vergleichende Operatoren und Bedingungen ..... | 5 |
| Funktionen .....                               | 6 |
| Variablen .....                                | 7 |
| Verkettungen .....                             | 8 |
| Überlaufzeichen .....                          | 8 |
| Wertebereiche .....                            | 8 |
| Kommentare .....                               | 9 |

# Cache-Variablen

## Einleitung

c:geo bietet für jede [Cache-Detail Ansicht](#) einen Karteireiter mit dem Namen "Variablen", der es ermöglicht Variablen und Formeln, die man für den Cache benötigt, zu notieren und Berechnungen mit ihnen durchzuführen.

The screenshot shows the 'Variables' tab of a cache detail view. At the top, there are buttons for adding variables (+), adding waypoints (+D), scanning a listing (+ SCAN LISTING), tidy up (TIDY UP), delete all (DELETE ALL), and help (HELP). Below these are three variable entries:

- A:** Formula  $f$   $2+2*5$ , Result  $= 12$ , Delete button (trash can icon).
- B:** Formula  $f$  `lettervalue('Cache')-12`, Result  $= 8$ , Delete button (trash can icon).
- C:** Formula  $f$  `ichecksum(1995)`, Result  $= 6$ , Delete button (trash can icon).

At the bottom of the screen, there are tabs for **VARIABLES (3)**, **WAYPOINTS (2)**, and **DETAILS**.

Dies kann praktisch sein, wenn du z.B. einen Multi-Cache suchst, der es erfordert draußen im Feld bestimmte

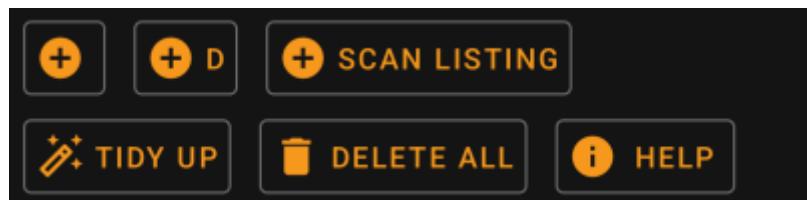
Werte zu sammeln und mit diesen mathematische Berechnungen durchzuführen um zur nächsten Station oder zum Final zu gelangen.

Du kannst diese Seite mit Variablen entweder nur für sich als Helfer für Berechnungen nutzen, oder du kannst auch jede Variable, die dort definiert ist in einem [berechneten Wegpunkt](#) für diesen Cache weiternutzen.

Der folgende Abschnitt dieser Seite beschreibt den Inhalt und die Funktionen der Variablen-Ansicht.

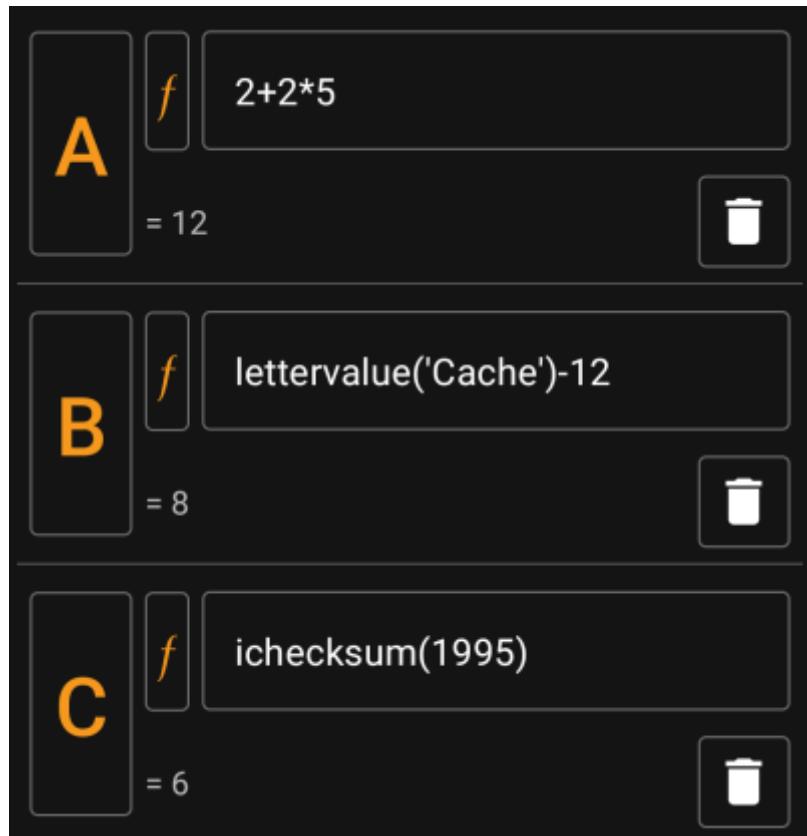
## Kontrollbereich

Oben in der Ansicht siehst du einige Schaltflächen, die dir Funktionen anbieten um den darunter befindlichen Variablenbereich zu befüllen:



| Schaltfläche | Beschreibung   |
|--------------|--|
|              | Eine selbst zu benennende Variable manuell zur Ansicht hinzufügen.   |
|              | Automatisch die nächste freie Variable (in alphabetischer Folge) zur Ansicht hinzufügen.   |
|              | Dieser Funktion scannt die Geocache-Beschreibung nach potentiellen Formeln und bietet an, diese in den Variablenbereich zu übernehmen. Jede ausgewählte gefundene Formel wird als Inhalt einer neuen Variable übernommen.        |
|              | Dies entfernt alle Variablen, die keinen Wert oder Formel enthalten. Die Funktion kann z.B. genutzt werden, wenn du aus Versehen zu viele Variablen angelegt hast oder du einige geleert hast, weil du sie nicht mehr benötigst. |
|              | Dies löscht alle angelegten Variablen und ihre Inhalte.  |
|              | Öffnet diese Seite im Browser.   |

## Variablenbereich



Dieser Bereich ermöglicht es einen Wert oder eine Formel für die erstellen Variablen einzugeben. Du kannst hier die folgenden Aktionen durchführen:

| Schaltfläche    | Aktion   |
|-----------------|--|
| A               | Tippe auf den Namen der Variable um ihn zu ändern.   |
| f               | Tippe auf diese "Funktions"-Schaltfläche, um das Wertefeld mit einer der <a href="#">unterstützten Funktionen</a> zu befüllen. |
| ichecksum(1995) | Trage hier manuell einen Wert oder eine Formel unter Nutzung der <a href="#">Formelsyntax</a> ein.                             |
| trash           | Nutze das Papierkorb-Symbol um die Variable zu löschen.  |

Der Text unter dem Variablenfeld zeigt eine Vorschau des Ergebnisses. Dies ist entweder das konkrete Ergebnis der Formel oder gibt Hinweise bzgl. Syntax-Fehlern oder fehlenden Werten.

## Formelsyntax

Das Wertefeld jeder Variable kann verschiedene Arten von Werten und auch andere Variablen enthalten. Es unterstützt vielfältige mathematische Operationen sowie einige (teilweise geocaching-bezogene) numerische und alphanumerische Funktionen, die im Folgenden beschrieben werden.



Fürchte dich nicht vor der Syntax. Sie unterstützt zwar auch relativ komplexe Operationen, can



aber auch für einfache Kalkulationen genutzt werden, wie du sie von jedem Taschenrechner kennst. Einige der Funktionen sind wahrscheinlich nur für fortgeschrittene Nutzer.

Die Syntax wird in den folgenden Unterkapiteln im Detail beschrieben. Als erste Übersicht über die unterstützten Funktionen, findest du hier eine Liste von Beispielen:

- `2*2+3` ergibt 7
- `2*(2+3)` ergibt 10
- `3*sin(90)` ergibt 3
- `4+length('test')` ergibt 8
- `rot13('abc')` ergibt nop
- `lettervalue('cache')` ergibt 20
- `checksum(lettervalue('cache'))` ergibt 2
- A + A\*2 with A=3 ergibt 9
- AA(A+1) with A=3 ergibt 334
- \$hello + 1 mit der Variable hello=24 ergibt 25
- \$hello(A+1) mit A=3 und hello=24 ergibt 244
- \${hello}8A mit A=3 und hello=24 ergibt 2483

## Typen von Variablen

Die Formelsyntax unterstützt drei Typen von Variablen. Du kannst einfach drauf los schreiben, generell versucht die Formelberechnung die gegebenen Werte so gut wie möglich einzupassen.

| Typ                     | Beschreibung               | Anwendung  | Bedeutung                           |
|-------------------------|----------------------------|--|-------------------------------------|
| Ganzzahlen (Integer)    | Zahlen ohne Dezimalstellen | Nutze Zahlen   | 1234, -3                            |
| Dezimalzahlen (Decimal) | Zahlen mit Dezimalstellen  | Nutze Zahlen mit Punkt oder Komma  | 3.14, -3.14, 3,14                   |
| Text (String)           | Text                       | Umgebe Text mit ' oder " Um die Symbole '... ' oder "..." selbst zu nutzen, gib ' ' oder " " ein | 'test', "test"<br>"Er sagt ""ja""!" |

## Numerische Operatoren

Die folgenden numerischen Operationen werden unterstützt:

| Operator | Funktion                                 | Beispiel   |
|----------|--|--|
| +        | Addition                                 | <code>2+4</code> ergibt 6                                  |
| -        | Subtraktion<br>(oder eine Zahl negieren) | <code>6-4</code> ergibt 2<br><code>-(5-2)</code> ergibt -3 |
| *        | Multiplikation                           | <code>3*4</code> ergibt 12                                 |
| /        | Division                                 | <code>12/3</code> ergibt 4                                 |
| %        | Modulo                                   | <code>12%5</code> ergibt 2                                 |
| ^        | Potenzieren                              | <code>3^3</code> ergibt 27                                 |
| !        | Faktorisieren                            | <code>4!</code> ergibt 24                                  |

## Vergleichende Operatoren und Bedingungen

Vergleichende Operatoren wie < or == können dazu verwendet werden, um zwei Werte miteinander zu

vergleichen. Generell ergibt eine solche Operation den Wert 1, wenn der Vergleich wahr ist und den Wert 0, wenn der Vergleich falsch ist.

Beispielsweise ergibt der Ausdruck `3 < 4` den Wert 1.

Vergleichende Operatoren werden insbesondere in der `if`-Funktion verwendet. Dieser Funktion berechnet ihren ersten Parameter. Wenn der Parameter wahr ist (bedeutet: Ergebnis ist `>0` oder ein nicht leerer String) gibt die Funktion ihren zweiten Parameter zurück. Im anderen Fall, und wenn sie einen dritten Parameter hat, wird dieser dritte Parameter zurückgegeben.

Die `if`-Funktion akzeptiert jede Anzahl von Parametern und interpretiert diese in einer "if-then-if-then-if-then-...-else"-Kaskade.

Dies bedeutet, dass z.B. wenn die Funktion 5 Parameter hat:

- \* Wenn der erste Parameter wahr ist, wird der zweite zurückgegeben
- \* Andernfalls, wenn der dritte Parameter war ist, wird der vierte Parameter zurückgegeben.
- \* Andernfalls wird der fünfte Parameter zurückgegeben.

Beispielsweise ergibt `if(A==5;50;A==4;40;30)` den Wert 50 wenn `A=5` ist, 40 wenn `A=4` ist und 30 für jeden anderen Wert von `A`.

| <b>Operator</b>       | <b>Bedeutung</b>            | <b>Beispiel</b>                         |
|-----------------------|-----------------------------|---|
| <code>==</code>       | Prüft Gleichheit            | <code>2==2</code> ergibt 1(=wahr)       |
| <code>&lt;&gt;</code> | Prüft Ungleichheit          | <code>3&lt;&gt;2</code> ergibt 1(=Wahr) |
| <code>&lt;</code>     | Ist kleiner als             | <code>3&lt;4</code> ergibt 1(=wahr)     |
| <code>≤</code>        | Ist kleiner oder gleich als | <code>3≤3</code> ergibt 1(=wahr)        |
| <code>&gt;</code>     | Ist größer als              | <code>3&gt;4</code> ergibt 0(=falsch)   |
| <code>≥</code>        | Ist größer oder gleich als  | <code>5≥5</code> ergibt 1(=wahr)        |

## Funktionen

Funktionen starten alle mit einem Buchstaben, enthalten nur Buchstaben oder Ziffern und haben eine direkt daran angeschlossene Liste von Parametern, die in Klammern stehen. Mehrere Parameter werden dabei durch ; getrennt.

Ein Beispiel für einen Funktionsaufruf mit einem Parameter ist `sin(90)`. Ein Beispiel für einen Funktionsaufruf mit zwei Parametern ist `rot('test'; 13)`.

Die folgenden Funktionen sind definiert:

| <b>Funktion</b>          | <b>Synonym</b> | <b>Beschreibung</b>                                      | <b>Parameter 1</b>               | <b>Parameter 2</b>                           | <b>Beispiel</b>   |
|--------------------------|----------------|--|----------------------------------|--|---|
| <code>sqrt</code>        | -              | Berechnet die Quadratwurzel des gegebenen Parameters     | Numerischer Parameter            | -  | <code>sqrt(9)</code> ergibt 3   |
| <code>sin/cos/tan</code> | -              | Berechnet Sinus/Cosinus/Tangens des gegebenen Parameters | Numerischer Parameter in Grad(!) | -  | <code>sin(90)</code> ergibt 1   |
| <code>abs</code>         | -              | Berechnet den absoluten Wert                             | Numerischer Parameter            | -  | <code>abs(-34)</code> ergibt 34   |
| <code>round</code>       | -              | Rundet Dezimalwerte mathematisch                         | Zu rundernder Wert               | Optional: Anzahl der Stellen für die Rundung | <code>round(4.65)</code> ergibt 5, <code>round(4.65; 1)</code> ergibt 4.7 |

| Funktion    | Synonym                 | Beschreibung  | Parameter 1   | Parameter 2  | Beispiel   |
|-------------|-------------------------|---|---|--|--|
| trunc       | -                       | Schneidet Dezimalwerte auf die (optional) angegebene Anzahl von Stellen   | Abzuschneidend Wert   | Optional: Anzahl Nachkommastellen, die nach dem Abschneiden verbleiben sollen (Standard:0) | trunc(123.456) ergibt 123,<br>trunc(123.456;2) ergibt 123.45 |
| if          | -                       | Wertet Bedingungen aus und gibt davon abhängige Werte zurück  | Liste von wenn-dann-sonst Werten. Siehe vorheriges Kapitel für Details. | -  | if(3<4;5;6) ergibt 5   |
| checksum    | cs                      | Berechnet die Quersumme des angegebenen numerischen Wertes. Berechnet den Buchstabenwert, wenn der angegebene Parameter ein Text ist                    | Positive Ganzzahlen oder Text   | -  | checksum(345) ergibt 12                                      |
| ichecksum   | ics                     | Berechnet die absolute/iterierte Quersumme des angegebenen numerischen Wertes. Startet beim Buchstabenwert, wenn der angegebene Parameter ein Text ist. | Positive Ganzzahl oder Text   | -  | ichecksum(345) ergibt 3                                      |
| lettervalue | lv,<br>wordvalue,<br>wv | Berechnet den Buchstabenwert des angegebenen Textes   | Text  | -  | lettervalue('test') ergibt 64                                |
| rot         | -                       | Berechnet einen rotierten Text aus dem angegebenen Text   | Text  | Anzahl der zu rotierenden Stellen  | rot('abc'; 1) ergibt 'bcd'                                   |
| rot13       | -                       | Berechnet den ROT-13 Wert aus dem angegebenen Text  | Text  | -  | rot13('abc') ergibt 'nop'                                    |
| roman       | -                       | Durchsucht einen gegebenen Text nach römischen Zahlen und gibt den dezimalen Wert zurück  | Text  | -  | roman('VI') ergibt 6.  |
| vanity      | vanitycode,<br>vc       | Gibt den Vanity-Code eines Textes zurück  | Text  | -  | vanity('cgeo') ergibt 2436.                                  |

## Variablen

Variablen werden in Formeln als Platzhalter für Werte verwendet. Wenn eine Formel berechnet wird, die eine Variable enthält, benötigt diese einen Wert für jede enthaltene Variable, damit sie korrekt berechnet wird.

Namen von Variablen unterscheiden Groß-/Kleinschreibung und müssen mit einem Buchstaben beginnen. Die restlichen Zeichen können Buchstaben oder Ziffern sein. Beispiele für gültige Variablennamen sind: Test, T1, t, Tt123. Beispiele für ungültige Variablennamen sind: 1a, 2.

Variablen, die nur aus einem Buchstaben bestehen, können einfach in Formeln eingegeben werden und werden direkt ausgewertet. Zum Beispiel ist die Formel A + 2 gültig. Wenn A den Wert 5 hat, ergibt die Formel 7.

Wenn mehrere Buchstaben in einer Formel direkt hintereinander geschrieben werden, werden diese als Wiederholung dieser Ein-Buchstaben-Variable angesehen. Zum Beispiel wird die Formel AA + 2 als zweifache

Verwendung der Variable A interpretiert und die 2 anschließend addiert. Wenn A=4 ist, ergibt diese Formel somit  $44 + 2 = 46$ . Das folgende Kapitel gibt mehr Informationen zum Thema Verketten von Variablen.

Namen von Variablen mit mehr als einem Buchstaben können im Unix-Bash-Stil verwendet werden, in dem ihnen ein \$ vorausgestellt wird. Beispielsweise kann eine Variable mit der Bezeichnung Test verwendet werden, in dem in Formeln \$Test eingesetzt wird. Die Formel \$Test + 2 ist somit gültig. Wenn die Variable Test gleich 4 ist, ergibt die Formel eine 6.

In Situationen, wo Variablennamen in Konflikt mit darauffolgenden Buchstaben stehen, können die Variablennamen in {} gesetzt werden, um diese vom folgenden Text zu trennen. Beispielsweise verkettet der folgende Ausdruck den Wert der Variable Test mit dem Wert der Variable A: \${Test}A

Einige weitere komplexe Beispiele:

- Die Formel A + \$A \* \$Test - t nutzt drei Variablen, nämlich A, Test und t. Angenommen A=2, Test=3 und t=1 ergibt die Formel 7.
- Die Formel AA + b + \$A1 nutzt drei Variablen, nämlich A, b und A1. Angenommen A=2, b=3 und A1=4 ergibt die Formel 29 (=  $22 + 3 + 4$ )
- Die Formel AB(B+1) nutzt zwei Variablen, nämlich A und B. Angenommen A=2 und B=3 ergibt die Formel 234
- Die Formel \$AB(B)(B+1) nutzt zwei Variablen, nämlich AB und B. Angenommen AB=2 und B=5 ergibt die Formel 256
- Nutzt man die {}-Syntax könnte das vorherige Beispiel auch wie folgt geschrieben werden:  
  \${AB}B(B+1)

## Verkettungen

Wenn mehrere Ausdrücke direkt hintereinander ohne trennenden Operator verkettet werden, werden deren Werte ebenso zu einem fortlaufenden Ausdruck verkettet. Dieser Ausdruck ergibt eine Zahl wenn es eine gültiger numerischer Ausdruck ist, sonst zu einem Textwert.

Ausdrücke, die verketten werden können, sind z.B. Ganzzahlen, Variablen, Ausdrücke in Klammern und das Überlaufzeichen (siehe nächstes Unterkapitel).

Beispielsweise enthält die Formel AA(A+4)55\$Test(3) zwei Variablen A und Test, Unter der Annahme, dass A=9 und Test=70 ist, ergibt sich das Ergebnis der Formel als 991355703.

## Überlaufzeichen

In verketteten Ausdrücken, kann das Zeichen \_ als Überlaufkennzeichnung verwendet werden. Es ist ein Platzhalter für mögliche Überläufe wenn eine numerische Variable einen Wert ergibt, der mehr als eine Stelle hat. Ansonsten wird das Platzhalter mit einer 0 gefüllt.

Ein Beispiel sollte die Nutzung klarer machen:

- Die Formel 1A mit A=2 ergibt 12
- Die Formel 1\_A mit A=2 ergibt 102
- Die Formel 1\_A mit A=23 ergibt 123
- Die Formel 1\_\_A mit A=23 ergibt 1023
- Die Formel 1\_\_A mit A=234 ergibt 1234

## Wertebereiche

Du kannst in Formeln Wertebereiche angeben, in dem du [] nutzt. Dies wird benötigt, wenn du Variablen über

einen bestimmten Wertebereich auswerten willst. Ein prominentes Beispiel ist die Funktion zur [Generierung von Wegpunkten](#).



Link to anchor on waypoint calc page as soon as its updated to cover waypoint generation with ranges.

Ein Beispiel für einen Wertebereich-Ausdruck ist `[0-9]`. Dies gibt einen Bereich mit 10 Werte (die Zahlenwerte 0 bis 9) an.

Du kannst mit dem Trennzeichen `,` fortlaufende Werte angeben. Du kannst Werte oder Wertebereiche durch Voranstellen von `^` ausschließen. Die Bereiche werden von links-nach-rechts gelesen und ergeben daraus die Anordnung der Elemente im Bereich. Die folgenden Beispiele zeigen gültige Bereichsangaben:

- `[0-2, 4]` ergibt einen Wertebereich, der 0, 1, 2 and 4 enthält.
- `[0-3, ^1-2]` ergibt einen Wertebereich, der 0 and 3 enthält.
- `[0-3, ^1-2, 5]` ergibt einen Wertebereich, der 0, 3 and 5 enthält.

Wenn ein Wertebereich in einem Kontext verwendet wird, in dem nur ein Einzelwert erlaubt ist (das ist bei normalen Berechnungen der Fall), wird der erste Wert im Bereich für die Berechnung genommen. Beispielsweise wird der Wertebereich `[0-9]` in normalen Berechnungen als 0 angenommen, während `[8, 0-9]` als 8 angenommen wird.

Wertebereiche unterstützen aktuell nur positive konstante Ganzzahlen. Ein Bereich muss immer mindestens einen Wert und nicht mehr als zwanzig Werte enthalten. Beispielsweise sind die folgenden Wertebereiche ungültig:

- `[]`: leer
- `[5, ^0-9]`: ergibt leer
- `[0-1000]`: ergibt mehr als zwanzig Werte
- `[-5]`: negative Zahlen nicht erlaubt
- `[A]`: Variablen nicht erlaubt

Eine Formel kann eine oder mehrere Wertebereiche gemischt mit normalen anderen Formelelementen enthalten. Zum Beispiel sind folgende Formeln gültig:

- `3*[0-2]`: ergibt die Werte 0, 3 und 6
- `A*[4, 7]`: Für A=3 ergibt dies die Werte 12 and 21
- `[1-2]*[3-4]`: ergibt 3, 6, 4 und 8.

## Kommentare

Durch Nutzung von `#` kannst du in Formeln auch Kommentare eintragen. Die Kommentare enden am nächsten `#` oder am Ende der Formel. Alles, was im Kommentar steht, wird während der Berechnung ignoriert. Beispiele:

- `A * 5 # Test Kommentar mit A=3 ergibt 15`
- `3.14 # Das ist Pi # * 2 # und das ist zwei ergibt 6.28`