Table of Contents

Cache Variabelen 2
Introductie 2
Bedieningsgedeelte 3
Variabele sectie 3
Formule syntaxis 4
Waardetypes 5
Numerieke operatoren 5
Relationele operatoren en voorwaarden 5
Functies 6
Variabelen 7
Aaneenschakelingen 8
Overloopteken 8
Bereikuitdrukkingen 8
Opmerkingen 9

Cache Variabelen

Introductie

Voor elke geocache detailweergave biedt c:geo een tabblad genaamd “Variabelen” om te noteren en berekeningen uit te voeren met formules en variabelen die je nodig hebt voor deze cache.

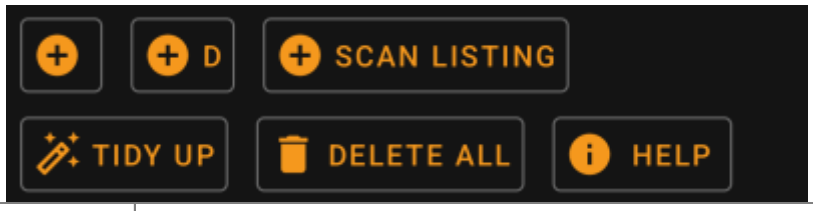
Dit kan handig zijn als je b.v. voor een multi-cache in het veld waarden moet verzamelen en hiermee wiskundige berekeningen moet uitvoeren om naar de volgende fase of de finale te gaan.

Je kunt dit tabblad met variabelen gebruiken als een op zichzelf staande helper om wat berekeningen uit te voeren of je kunt ook elke hier gedefinieerde variabele hergebruiken voor het genereren van een nieuw berekend waypoint voor deze cache.

De volgende secties op deze pagina beschrijven de inhoud en functionaliteit van het variabele tabblad.

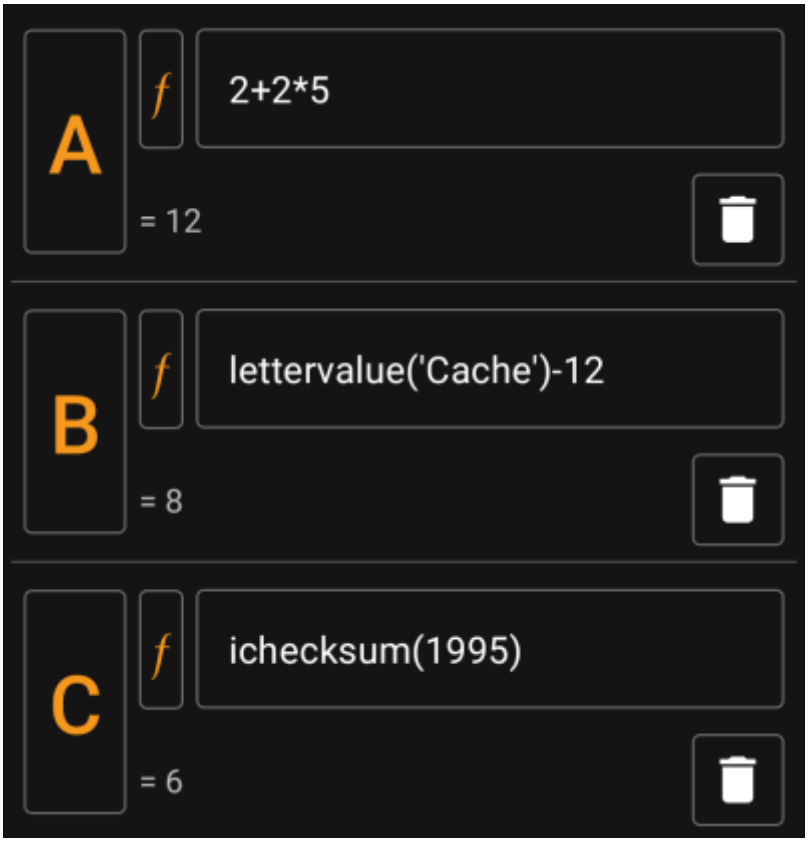
Bedieningsgedeelte

Boven aan het variabele tabblad zie je een reeks knoppen die functies bieden om het variabele gedeelte hieronder te vullen:



Knop	Beschrijving
	Voeg handmatig een variabele toe aan de weergave door de naam op te geven.
	Voeg automatisch de volgende vrije variabele in alfabetische volgorde toe aan de weergave.
	Deze functie scant de geocachebeschrijving op mogelijke formules die erin zitten en biedt ze aan om ze over te nemen naar het tabblad met variabelen. Elke geselecteerde gevonden formule wordt toegevoegd als inhoud van een nieuwe variabele.
	Hiermee worden alle variabelen zonder waarde of formule verwijderd en kan worden gebruikt voor het geval je per ongeluk veel variabelen hebt aangemaakt of sommige ervan hebt gewist en ze niet langer nodig hebt.
	Hiermee worden alle gedefinieerde variabelen en hun waarden verwijderd.
	Opent deze pagina in je browser

Variabele sectie



In deze sectie kun je de waarde of formule voor de gegenereerde variabelen invoeren. Je kunt hier de volgende acties uitvoeren:

Knop	Actie
	Klik op de naam van de variabele om deze te wijzigen.
	Klik op de “functie”-knop om het waardeveld vooraf in te vullen met de gewenste ondersteunde functie.
	Vul het waardeveld handmatig in met een waarde of een formule met behulp van de formulesyntaxis.
	Gebruik het prullenbakpictogram om de variabele te verwijderen.

De tekst onder het waardeveld toont een voorbeeld van een resultaat. Dit kan het concrete resultaat van de formule zijn of hints met betrekking tot syntaxisfouten of ontbrekende waarden.

Formule syntaxis

Het waardeveld van elke variabele kan verschillende soorten waarden en ook andere variabelen bevatten. Het ondersteunt tal van wiskundige bewerkingen, evenals verschillende (deels aan geocaching gerelateerde) numerieke en tekenreeksgerelateerde functies, zoals hieronder beschreven.

Wees niet bang voor de syntaxis. Hoewel het vrij complexe bewerkingen ondersteunt, kan het ook

worden gebruikt voor eenvoudige en duidelijke berekeningen, zoals je van elke rekenmachine gewend bent. Sommige ondersteunde functies zijn waarschijnlijk alleen voor gevorderde gebruikers.

De syntaxis wordt in de volgende subhoofdstukken in detail uitgelegd. Hier zie je lijst met voorbeelden van wat wordt ondersteund.

- `2*2+3` wordt geëvalueerd tot 7
- `2*(2+3)` zal evalueren tot `10`
- `3*sin(90)` zal evalueren tot 3

- `4+length('test')` wordt geëvalueerd tot 8
- `rot13('abc')` zal evalueren tot `nop`
- `lettervalue('cache')` wordt geëvalueerd tot `20`
- `checksum(lettervalue('cache'))` wordt geëvalueerd tot 2

- `A + A*2` met `A=3` wordt geëvalueerd tot 9
- `AA(A+1)` met `A=3` wordt geëvalueerd tot 334
- `$hello + 1` met variabele `hello=24` zal evalueren tot 25
- `$hello(A+1)` met `A=3` en `hello=24` wordt geëvalueerd tot 244
- `${hello}8A` met `A=3` en `hello=24` wordt geëvalueerd tot 2483

Waardetypes

De formulesyntaxis ondersteunt drie typen waarden. Typen is een ruim begrip, in het algemeen zal de evaluatie van de formule proberen de gegeven waarden zo goed mogelijk te laten passen.

Type	Beschrijving	Letterlijke syntaxis	Voorbeelden
Heel getal	Getal zonder decimalen	Gebruik cijfers	`1234`, `-3`
Decimaal	Getal met decimalen	Gebruik cijfers met decimale punt of komma	`3.14`, `-3.14`, `3,14`
String	Tekst	Waarde omringen met `'` of `"` Om de `'...'` of `"..."` zelf te gebruiken, typ `''` of `""`	`'test'`, `"test"` `“Hij zei ”“ja”“!”`

Numerieke operatoren

De volgende Numerieke operatoren worden ondersteund:

Operator	Functie	Voorbeelden
+	Optellen	2+4 evalueert naar 6
-	Aftrekken	6-4 evalueert naar 2 `-(5-2)` evalueert naar `-3`
*	Vermenigvuldigen	3*4 evalueert naar 12
/	Delen	12/3 evalueert naar 4
%	Modulo	12%5 evalueert naar 2
^	Machtsverheffen	3^3 evalueert naar 9
!	Factorisatie	`4!` evalueert naar 24

Relationele operatoren en voorwaarden

Relationele operatoren zoals < of == kunnen worden gebruikt om twee waarden met elkaar te vergelijken. In het algemeen zal een dergelijke bewerking de waarde 1 retourneren als de vergelijking waar oplevert en de waarde `0` als het onwaar oplevert.

De uitdrukking `3 < 4` wordt bijvoorbeeld berekend tot de waarde 1.

Relationele operatoren worden vooral gebruikt in de `if`-functie. Deze functie evalueert zijn eerste parameter. Als deze parameter waar is (betekent: heeft een waarde > 0 of is een niet-lege tekenreeks), dan retourneert deze de tweede parameter. Anders, en als het een derde parameter heeft, wordt de derde parameter geretourneerd.

De `if`-functie accepteert een willekeurig aantal parameters en interpreteert ze in een “als-dan-als-als-dan-...-anders” cascade.

Dit betekent dat als de functie 5 parameters heeft gekregen, dan: * Als de eerste parameter waar is, wordt de tweede geretourneerd * Anders, als de derde parameter waar is, wordt de vierde parameter geretourneerd * Anders wordt de vijfde parameter geretourneerd

Bijvoorbeeld `if(A==5;50;A==4;40;30)` wordt geëvalueerd tot `50` als `A=5`, tot `40` als `A=4` en naar `30` voor elke andere waarde van `A`.

Operator	Betekenis	Voorbeeld
==	Controles op gelijkheid	2==2 resulteert in 1(=true)
<>	Controleert op ongelijkheid.	3<>2 resulteert in 1(=true)
<	Is kleiner dan	3<4 resulteert in 1(=true)
⇐	Is kleiner of gelijk aan	3⇐3 resulteert in 1(=true)
>	Is groter dan	3>4 resulteert in 0(=false)
>=	Is groter of gelijk aan	5>=5 resulteert in 1(=true)

Functies

Functies beginnen allemaal met een letter, bevatten alleen letters en cijfers en hebben een direct gekoppelde parameterlijst tussen haakjes. Meerdere parameters worden gescheiden met `;`.

Een voorbeeld van een functieaanroep met één parameter is `sin(90)`. Een voorbeeld van een functieaanroep met twee parameters is `rot('test'; 13)`.

De volgende functies zijn gedefinieerd:

Functie	Synoniemen	Beschrijving	Parameter 1	Parameter 2	Voorbeeld
`sqrt`	-	berekent vierkantswortel van gegeven parameter	numerieke parameter	-	`sqrt(9)` evalueert naar 3
`sin/cos/tan`	-	berekent sinus/cosinus/tangens van gegeven parameter	numerieke parameter in graden(!)	-	`sin(90)` evalueert tot 1
`abs`	-	berekent absolute waarde	numerieke parameter	-	`abs(-34)` evalueert tot 34
`round`	-	rondt decimale waarden wiskundig af	waarde om af te ronden	optioneel: aantal plaatsen om op af te ronden	`round(4.65)` evalueert tot 5, `round(4.65;1)` evalueert tot `4.7`
`trunc`	-	Kapt decimale waarden af tot het (optioneel) opgegeven aantal plaatsen	waarde om af te kappen	optioneel: het aantal resterende plaatsen na afkappen (standaard: 0)	`trunc(123.456)` wordt 123, `trunc(123.456;2)` wordt `123.45`

Functie	Synoniemen	Beschrijving	Parameter 1	Parameter 2	Voorbeeld
`if`	-	evalueert voorwaarden en retourneert voorwaardelijke waarden	lijst met als-dan-anders-waarden. Zie vorige sectie voor details	-	`if(3<4;5;6)` evalueert tot 5
`checksum`	`cs`	berekent de controlesom van de gegeven numerieke waarde. Berekent letterwaarde als gegeven parameter van het type tekst is	positief geheel getal of tekst	-	`checksum(345)` evalueert tot 12
`ichecksum`	`ics`	berekent iteratieve controlesom van een gegeven numerieke waarde. Begint vanaf letterwaarde als de gegeven parameter van het type tekst is	positief geheel getal of tekst	-	`ichecksum(345)` evalueert tot 3
`lettervalue`	`lv`, `wordvalue`, `wv`	berekent letterwaarde van gegeven stringwaarde	tekenreeks	-	`lettervalue('test')` evalueert tot 64
`rot`	-	berekent geroteerde string van gegeven stringwaarde	tekenreeks	tel om te roteren met	`rot('abc'; 1)` evalueert naar `'bcd'`
`rot13`	-	berekent gedraaid-13 van gegeven stringwaarde	tekenreeks	-	`rot13('abc')` evalueert tot `'nop'`
`roman`	-	scant een gegeven tekenreekswaarde als een Romeins getal en retourneert de decimale waarde	tekenreeks	-	`roman('VI')` evalueert tot 6.
`vanity`	`vanitycode`, `vc`	geeft de vanity-code van een string terug	tekenreeks	-	`vanity('cgeo')` evalueert tot 2436.

Variabelen

Variabelen worden in een formule gebruikt als tijdelijke aanduidingen voor waarden. Wanneer een formule die een variabele bevat, wordt geëvalueerd, moet er een waarde aan worden doorgegeven voor elk van de opgenomen variabelen om correct te kunnen worden geëvalueerd.

Namen van variabelen zijn hoofdlettergevoelig en moeten beginnen met een alfanumeriek teken. Resterende tekens kunnen alfanumeriek of cijfers zijn. Voorbeelden voor namen van wettelijke variabelen zijn: `Test`, `T1`, `t`, `Tt123`. Voorbeelden van namen van niet-wettelijke variabelen zijn: `1a`, `2`

Variabelen van één letter kunnen gewoon in de formule worden getypt en worden mee geëvalueerd. Zo is de formule `A + 2` geldig. Als A de waarde 5 heeft, wordt de formule geëvalueerd tot 7.

Als meerdere tekens binnen een formule worden aaneengeschakeld, worden ze geïnterpreteerd als afzonderlijke éénlettervariabelen. De formule `AA + 2` wordt bijvoorbeeld geïnterpreteerd als variabele A, twee keer aaneengeschakeld en daarna 2. Als A=4, zal deze formule resulteren in `4 + 2 = 46`. Zie de volgende sectie voor meer details tov aaneenschakeling.

Variabelenamen die langer zijn dan één char kunnen in Unix-Bash-Style gedeclareerd worden door hun naam vooraf te laten gaan aan `$`. Er kan bijvoorbeeld naar een variabele met de naam `Test` worden verwezen met

`$Test`. De formule `$Test + 2` is geldig. Als de waarde voor variabele `Test` 4 is, wordt de formule geëvalueerd tot 6.

In situaties waarin de naam van de variabele in strijd is met de volgende alfa's/tekens, kan de naam van de variabele worden ingesloten in `{}` om deze te onderscheiden van de volgende tekst. De volgende uitdrukking zal bijvoorbeeld de waarde van variabele `Test` samenvoegen met de waarde van variabele A: `${Test}A`

Enkele meer complexe voorbeelden:

- De formule `A + $A * $Test - t` gebruikt drie variabelen genaamd A, `Test` en `t`. De variabele A wordt op twee plaatsen gebruikt. Uitgaande van A=2, `Test`=3 en `t`=1, zou de formule resulteren in 7.
- De formule `AA + b + $A1` gebruikt drie variabelen A, b en A1. Uitgaande van A=2, b=3 en A1=4, zou de formule resulteren in 29 (= 22 + 3 + 4)
- De formule `AB(B+1)` gebruikt twee variabelen A en B. Uitgaande van A=2 en B=3, zou de formule resulteren in 234
- De formule `$AB(B)(B+1)` gebruikt twee variabelen AB en B. Uitgaande van AB=2 en B=5, zou de formule resulteren in 256
- Met de syntaxis `{}` kan het vorige voorbeeld ook als volgt worden geschreven: `${AB}B(B+1)`

Aaneenschakelingen

Als meerdere uitdrukkingen direct na elkaar worden aaneengeschakeld zonder scheidingsoperator, worden waarden aaneengeschakeld tot een opeenvolgende uitdrukking. Deze uitdrukking evalueert tot een getal als het een geldige numerieke uitdrukking vormt, anders evalueert het tot een tekstwaarde.

Expressies, die aaneengeschakeld kunnen worden, omvatten b.v. gehele cijfers, variabelen, uitdrukkingen tussen haakjes en het Overloop-teken (zie volgende paragraaf).

De formule `AA(A+4)55$Test(3)` bevat bijvoorbeeld twee variabelen A en `Test`. Uitgaande van A=9 en `Test=70`, zou dit resulteren in `991355703`.

Overloopteken

In aaneengeschakelde uitdrukkingen kan het teken _ worden gebruikt als overloopteken. Het is een tijdelijke aanduiding voor mogelijke overloopeffecten als numerieke variabelen resulteren in een waarde met meer dan één cijfer, anders wordt het gevuld met 0.

Een voorbeeld zou het gebruik duidelijk moeten maken:

- De formule 1A met A=2 zal resulteren in 12
- De formule 1_A met A=2 zal resulteren in 102
- De formule 1_A met A=23 zal resulteren in 123
- De formule 1__A met A=23 zal resulteren in 1023
- De formule 1__A met A=234 zal resulteren in 1234

Bereikuitdrukkingen

Je kunt bereiken in formules specificeren met `[]`. Dit is nodig wanneer variabelen worden gebruikt in een context waarin een reeks waarden moet worden herhaald. Een prominent voorbeeld is de functie Genereer Waypoints.

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

Een voorbeeld voor een bereikuitdrukking is `[0-9]`. Dit specificeert een bereik met 10 waarden (de integerwaarden 0 tot 9).

Je kunt opeenvolgende waarden opgeven met `,` als scheidingsteken. Je kunt waarden of waardebereiken uitsluiten door er een `^` aan toe te voegen. Bereiken worden van links naar rechts ontleed, waardoor een volgorde wordt gegeven aan de elementen in het bereik. De volgende zijn bijvoorbeeld geldige bereikspecificaties:

- `[0-2, 4]` resulteert in een bereik dat `0`, 1, 2 en 4 bevat.
- `[0-3, ^1-2]` resulteert in een bereik dat `0` en 3 bevat.
- `[0-3, ^1-2, 5]` wordt geëvalueerd tot een bereik dat `0`, 3 en 5 bevat.

Wanneer een bereik wordt gebruikt in een context waarin slechts één waarde is toegestaan (dit is het geval bij normale berekeningen), wordt de eerste bereikwaarde gebruikt voor de berekening. Bijvoorbeeld, de uitdrukking `[0-9]` zal evalueren tot `0` in een normale berekeningscontext, terwijl `[8, 0-9]` zal evalueren tot 8.

Bereiken ondersteunen momenteel alleen positieve constante gehele getallen. Een bereik moet altijd worden geëvalueerd tot ten minste 1 waarde en een bereik mag niet worden geëvalueerd tot meer dan 20 waarden. De volgende bereiken zijn bijvoorbeeld ongeldig:

- `[]`: leeg
- `[5, ^0-9]`: evalueert naar leeg
- `[0-1000]`: evalueert tot meer dan 20 inzendingen
- `[-5]`: negatieve int niet toegestaan
- `[A]`: variabelen niet toegestaan

Een formule kan een of meer bereikdefinities bevatten, gemengd met normale andere formuleonderdelen. De volgende formules zijn bijvoorbeeld geldig:

- `3*[0-2]`: evalueert tot waarden `0`, 3 en 6
- `A*[4, 7]`: voor A=3 resulteert dit in de waarden 12 en 21
- `[1-2]*[3-4]`: evalueert tot 3, 6, 4 en 8.

Opmerkingen

Je kunt opmerkingen in formule-uitdrukkingen invoeren met het teken #. Opmerkingen eindigen bij de volgende # of aan het einde van uitdrukkingen. Alles in een opmerking wordt tijdens de evaluatie genegeerd. Bijvoorbeeld:

- `A * 5 # testcommentaar` voor A=3 evalueert tot 15
- `3.14 # dit is pi # * 2 # en dit is twee` evalueert tot `6.28`