Mega-Event in Horn

MEGA

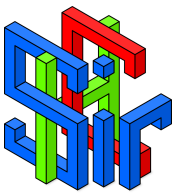
Waldviertel Geschichte und Geschichten GCA501G

c:geo als Rechenknecht

Nie wieder verrechnen bei Multis

Sir-h-c

8. Juni 2024



Inhaltsverzeichnis

1. Einleitung

Der Wetterbericht verspricht perfektes Geocaching-Wetter. Ein schöner Multi-Cache lockt zu einer aussichtsreichen Wanderung. Doch bald vergeht der Spaß, denn bei jeder Stage muss mühsam zur nächsten Stage gerechnet werden. Verrechner kosten Nerven, Zeit und teilweise erhebliche Umwege. Manchmal bleibt in letzter Konsequenz nur der Abbruch, ein DNF und ein langes Gesicht.

Damit ist jetzt Schluss. Die Android-App c:geo unterstützt GeocacherInnen mit hilfreichen Werkzeugen bei der Planung und Absolvierung von (Multi-)Caches, indem es die rechenintensive Arbeit abnimmt. Um das zu erreichen, ist etwas Einarbeitung erforderlich. Damit der Einstieg schnell gelingt, zeigt dieses Skriptum anhand ausgewählter Beispiele die Herangehensweise und mögliche Rechenarten bei typischen Multi- und Mystery-Caches.

Dieses Skriptum richtet sich an GeocacherInnen, die bereits Multi- und Mystery-Caches absolviert haben und im Umgang mit c:geo grundsätzlich vertraut sind. Nicht Teil des Skriptums ist das Installieren und Konfigurieren von c:geo sowie das Herunterladen und Verwalten von Geocaches.

2. Wegpunkte

Beim Geocaching haben wir es mit unterschiedlichen Typen von Wegpunkten zu tun. Multi-Caches haben einzelne Stationen oder Stages und zusätzlich etwa Referenzpunkte. Es können aber auch ein Parkplatz für das Cachemobil und ein Ausgangspunkt angegeben sein. Schließlich gibt es den ersehnten finalen Wegpunkt, an dem sich der Geocache befindet. Die in c:geo unterstützten Wegpunkt-Typen können unter https://manual.cgeo.org/de/cachedetails#wegpunkttyp nachgeschlagen werden.

In c:geo werden Wegpunkte auf dem Reiter „Wegpunkte“ organisiert. Auf diesem finden wir die vom Geocache-Owner vorgegebenen Wegpunkte und können eigene Wegpunkte hinzufügen. Eigene Wegpunkte werden auf Basis von Koordinaten, durch Projektion, durch Verschiebung oder basierend auf einer beliebigen Berechnung erstellt.

2.1. Koordinaten eingeben

Nachdem wir einen neuen Wegpunkt erstellt haben, können wir dessen Koordinaten eingeben. Ein Klick auf das Koordinatenfeld öffnet den Dialog zur Koordinateneingabe, wie in Abbildung 1 zu sehen ist. Durch einen Klick auf das Feld Koordinatenformat können

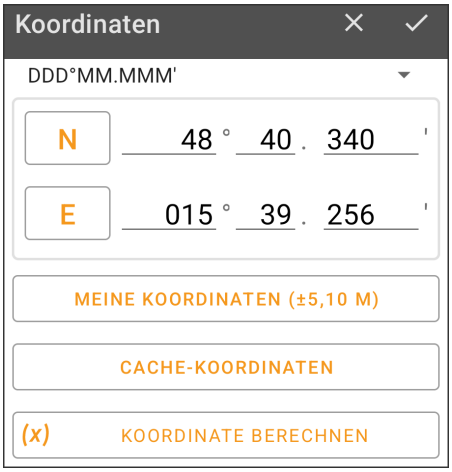


Abbildung 1: Koordinaten eines Wegpunkts eingeben

wir eines der Eingabeformate „Schlicht“, „DDD.DDDDD°“, „DDD°MM.MMM′“ oder „DDD°MM′SS.SSS″“ wählen. Das Eingabeformat hat keinen Einfluss auf das spätere Anzeigeformat der Koordinaten.

Details zum Dialog für die Koordinateneingabe sind im Benutzerhandbuch unter https://manual.cgeo.org/de/coordinatedialog nachzulesen.

2.2. Koordinaten-Projektion und -Verschiebung

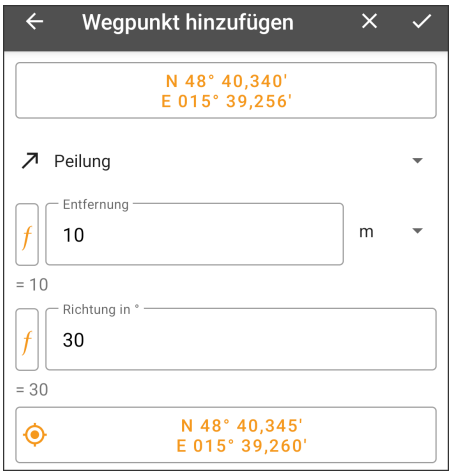


Abbildung 2: Koordinaten-Projektion eines Wegpunkts

Im vorigen Unterabschnitt 2.1 haben wir einen Wegpunkt erstellt. Ausgehend davon

können wir die Felder „Richtung in °“ und „Entfernung“ nutzen, um eine Koordinaten-Projektion durchzuführen, siehe Abbildung 2. c:geo zeigt die Ergebniskoordinaten an und übernimmt beim Speichern die neuen Koordinaten.

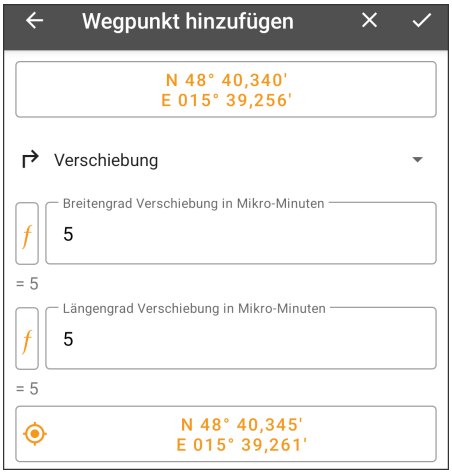


Abbildung 3: Koordinaten-Verschiebung eines Wegpunkts

Im Unterabschnitt 2.1 haben wir einen Wegpunkt erstellt. Ausgehend davon können wir durch Angabe von „Breitengrad“ und „Längengrad“ in Milliminuten eine Koordinaten-Verschiebung durchführen, siehe Abbildung 3. c:geo zeigt die Ergebniskoordinaten an und übernimmt beim Speichern die neuen Koordinaten.

2.3. Koordinaten berechnen

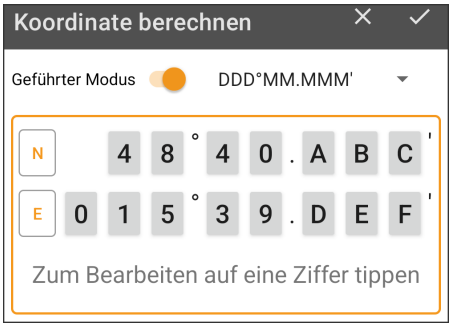


Abbildung 4: Geführter Modus, Ziffern durch Variablen ersetzt

Der Dialog Koordinateneingabe aus dem Unterabschnitt 2.1 erlaubt uns auch das Berechnen von Koordinaten. Dazu gibt es einen geführten Modus, der sich für Ersetzungen von einzelnen Ziffern eignet. Praktisch angewendet werden kann das beispielsweise beim Geocache „Schulstadt Horn“ (https://coord.info/GC5PRRP). Durch Tippen auf

ein Ziffernfeld kann eine Ziffer durch eine Variable ersetzt werden, wie Abbildung 4 zeigt. Sobald anschließend alle Variablen mit Werten befüllt wurden, berechnet c:geo die Koordinaten des Wegpunkts.

Abbildung 5: Problematische Verwendung von Variablen im geführten Modus

Obwohl es dieser Modus auch erlaubt, die drei Ziffern der Dezimalminuten durch eine einzige Variable zu ersetzen, ist davon dringend abzuraten. Abbildung 5 zeigt, wie c:geo in solch einem Fall mit warnenden Rufzeichen reagiert und zum Prüfen des Ergebnisses mahnt.

Neben dem geführten gibt es auch den freien Modus. Dieser erlaubt es uns, beliebig komplexe, mathematische Berechnungen durchzuführen. Je nach Art der Koordinatenmanipulation kann es erforderlich sein, gezielt Klammern zu setzen. Abbildung 6 zeigt das anhand eines Beispiels. Bei der Nordkoordinate wird die Klammer nach dem Komma der Dezimalminute gesetzt, wodurch die Addition des Werts der Variable „$A“ den Wert nach dem Komma ändert. Bei der Ostkoordinate hingegen wird die Klammer vor dem Wert der Dezimalminute gesetzt, wodurch der Wert der Variable „$B“ zum ganzzahligen Wert der Dezimalminuten addiert wird.

Zum Zeitpunkt des Erstellens dieses Skriptums im Juni 2024 ist das Berechnen von Koordinaten mittels Wegpunkt-Rechner im Benutzerhandbuch noch nicht korrekt dokumentiert. Die Beschreibung unter https://manual.cgeo.org/de/waypointcalculator sollte daher mit Vorsicht gelesen werden. Im weiteren Verlauf des Skriptums lernen wir Cache-Variablen kennen, mit deren Hilfe die hier beschriebenen Möglichkeiten erheblich erweitert werden.

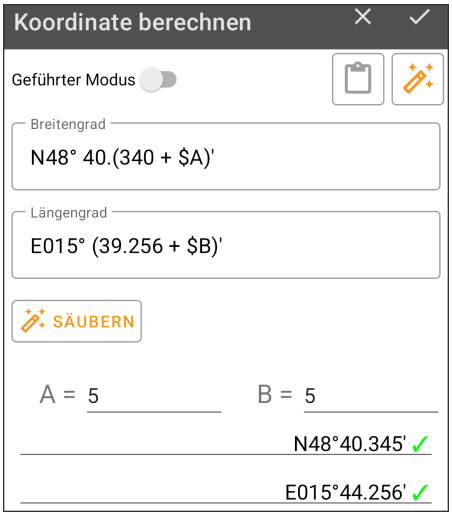


Abbildung 6: Freier Modus, gezielte Klammersetzung vor bzw. nach dem Komma

3. Variablen

Ab c:geo Version 2022.06.06[1] – also nach dem 6. Juni 2022 – gibt es einen überarbeiteten Wegpunkt- bzw. Koordinaten-Rechner. Die Variablen wurden auf eine eigene Variablen-Registerkarte ausgelagert, die nun auch die Nutzung erweiterter Funktionen ermöglicht.

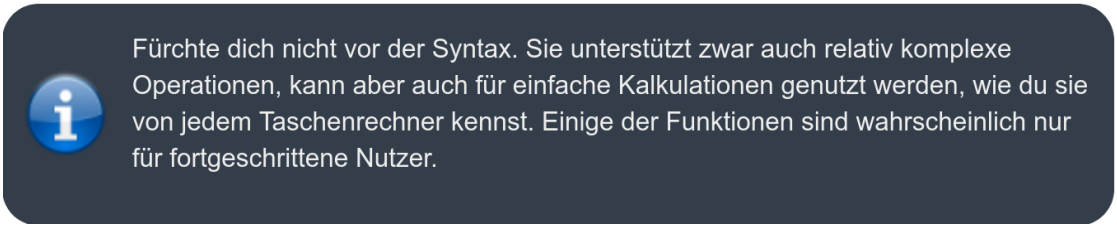


Abbildung 7: Information zur Formelsyntax

Im Benutzerhandbuch findet sich unter https://manual.cgeo.org/de/cachevariables derzeit eine Informationsbox, die darauf hinweist, sich nicht vor der Formelsyntax zu fürchten, siehe Abbildung 7. Die Verwendung von Funktionen kann anfangs ungewohnt sein. Um den Einstieg zu erleichtern, wollen wir uns in den folgenden Abschnitten mit Variablen, Funktionen und der Formelsyntax näher befassen.

[1] https://github.com/cgeo/cgeo/releases

3.1. Typen

Typ	Beschreibung	Beispiele
Long	Ganzzahl	1234 −42
Double	Dezimalzahl	3, 14 −3.14
String[2]	Text	"FTF" 'DNF'

Tabelle 1: Typen von Variablen

c:geo kennt im Wesentlichen drei Typen von Variablen: Positive und negative ganze Zahlen, Dezimalzahlen mit Dezimaltrennzeichen sowie Text, der in einfachen oder doppelten Anführungszeichen eingegeben wird. Tabelle 1 fasst die drei Typen zusammen und zeigt einige Beispiele.

c:geo versucht, eingegebene Werte so gut wie möglich in einen dieser Typen einzupassen. Bei komplexen, verschachtelten Berechnungen oder in Sonderfällen kann es erforderlich sein, insbesondere Text mit Anführungszeichen als solchen zu kennzeichnen, um ihn von der Interpretation als Name einer Variablen zu unterscheiden.

Im Abschnitt „Typen von Variablen“ im Benutzerhandbuch gibt es weitere Informationen zu den Typen, siehe https://manual.cgeo.org/de/cachevariables#typen_von_variablen.

3.2. Namen

Variablen werden mittels Schaltfläche im Kontrollbereich des Variablen-Reiters oder beim Einfügen in eine Formel angelegt. Variablennamen müssen aus mindestens einem Buchstaben bestehen (z.B. „A“). Namen von Variablen können auch aus mehreren Buchstaben gebildet werden und dann auch Zahlen enthalten, z.B. „Summe“ oder „Stage1“.

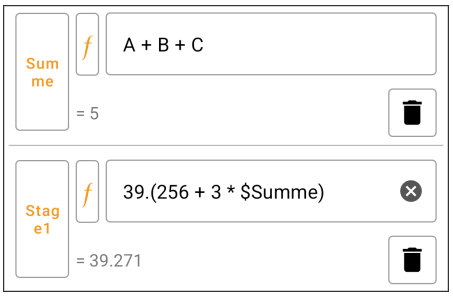


Abbildung 8: Lange Variablenname müssen mit Dollarzeichen verwendet werden

[2]Sonderfall: ”Grimm’sche Märchen” oder ’Ohm’ ’sches Gesetz’

Möchten wir eine Variable in einer Formel verwenden, muss dieser ein Dollarzeichen „$“ vorangestellt werden (z.B. „$Summe“). Das ist erforderlich, um Variablen von gewöhnlichem Text zu unterscheiden. Von dieser Regel gibt es eine Ausnahme: Variablennamen mit nur einem Buchstaben können auch ohne Dollarzeichen verwendet werden, vgl. Abbildung 8. Das ist praktisch, wenn Formeln aus dem Geocache-Listing kopiert werden.

Das Benutzerhandbuch erklärt anhand von Beispielen detailliert die einzelnen Situationen, siehe https://manual.cgeo.org/de/cachevariables#variablen.

3.3. Verkettung

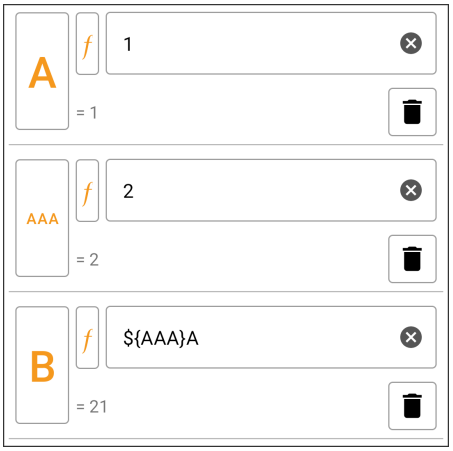


Abbildung 9: Verkettung von Variablen

Variablen können z.B. mit anderen Variablen, Ganzzahlen oder Ausdrücken in Klammern verkettet werden. Anstelle des Namens der Variable bzw. des Klammerausdrucks wird deren Wert mit den benachbarten Ausdrücken verkettet. Abbildung 9 zeigt die Verkettung der Variablen „AAA“ mit der Variablen „A“. Um die Variable „AAA“ von einer dreifachen Verkettung der Variablen „A“ zu unterscheiden, muss das Dollarzeichen verwendet werden. Um die Variable „$AAA“ mit verketteter Variable „A“ von der Variable „$AAAA“ zu unterscheiden, muss eine geschweifte Klammer gesetzt werden. Nur dann kann c:geo die Variablen richtig verarbeiten und verketten.

Weitere Details zu Verkettungen im Benutzerhandbuch unter https://manual.cgeo.org/de/cachevariables#verkettungen.

3.4. Numerische Operatoren

c:geo beherrscht die üblichen Rechenoperationen, die auch Taschenrechner-Apps können. Im Benutzerhandbuch fehlt allerdings, dass c:geo flexibel mit aus Geocache-Listings kopierten Formeln umgehen kann. Beispielsweise wird sowohl der Bindestrich „-“ als auch das längere mathematische Minuszeichen „−“ als Subtraktion erkannt. Ebenso wird die Multiplikation entweder durch das Sternsymbol „∗“ oder den Mittelpunkt „·“ durchgeführt. Die Division kann entweder durch Schrägstrich „/“, Doppelpunkt „:“ oder Divisonssymbol „÷“ angegeben sein. Potenzieren schließlich kann c:geo entweder mit hochgestellten Quadrat- und Kubikzahlen „x^2, x^3“ oder allgemein mit dem Dachsymbol „^“. Tabelle 2 listet die verfügbaren Operatoren mit ihren alternativen Symbolen auf und gibt kurze Beispiele zur Verwendung.

Operator	**Alternativ**	**Funktion**	**Beispiel**
+		Addition	2 + 4 ergibt 6
-	−	Subtraktion / Negation	6 - 4 ergibt 2
∗	·	Multiplikation	7 ∗ 6 ergibt 42
/	: ÷	Division	12/3 ergibt 4
%		Modulo	12 % 5 ergibt 2
x^2, x^3		Quadrat, Kubik	2^3 ergibt 8
^		Potenzieren	2^4 ergibt 16
!		Faktorisieren	4! ergibt 24

Tabelle 2: Numerische Operatoren

Im Benutzerhandbuch ist die Tabelle unter https://manual.cgeo.org/de/cachevariables#numerische_operatoren zu finden.

3.5. Vergleichende Operatoren

Um zwei Variablen bzw. deren Werte miteinander zu vergleichen, können vergleichende Operatoren verwendet werden. Die verfügbaren Operatoren zeigt Tabelle 3. Als Ergebnis dieser Operationen erhält man 1 wenn der Vergleich wahr oder 0 wenn der Vergleich falsch ist. Diese Operatoren werden insbesondere mit der, in Unterabschnitt 3.6 beschriebenen, „if“-Funktion verwendet. Ein Anwendungsfall ist z.B. der Vergleich einer berechneten Prüfsumme mit dem Sollwert aus dem Geocache-Listing. Dazu werden wir uns später in Abschnitt 6 praktische Beispiele ansehen.

Der zugehörige Abschnitt https://manual.cgeo.org/de/cachevariables#vergleichende_operatoren_und_bedingungen im Benutzerhandbuch beschreibt die Operatoren und deren Verwendung.

Operator	Bedeutung	Beispiel
==	Prüft Gleichheit	2 == 2 ergibt 1 (=wahr)
<>	Prüft Ungleichheit	3 <> 2 ergibt 1 (=wahr)
<	Ist kleiner als	3 < 4 ergibt 1 (=wahr)
<=	Ist kleiner oder gleich als	3 <= 3 ergibt 1 (=wahr)
>	Ist größer als	3 > 4 ergibt 0 (=falsch)
>=	Ist größer oder gleich als	5 >= 5 ergibt 1 (=wahr)

Tabelle 3: Vergleichende Operatoren

3.6. Funktionen

Funktionen erweitern c:geo um die Möglichkeit, beim Geocaching typischerweise benötigte Berechnungen oder Konvertierungen durchzuführen. Dazu zählen beispielsweise das Berechnen der Quersumme einer Zahl oder das Bilden des Buchstabenwortwerts.

Funktion	trunc, tr, floor, fl
Beschreibung	Schneidet Dezimalwerte auf angegebene Anzahl von Stellen ab.
Beispiel	tr(12,3) ergibt 12, tr(12,345;2) ergibt 12,34
Funktion	abs
Beschreibung	Berechnet den absoluten Wert.
Beispiel	abs(-42) ergibt 42
Funktion	sin/cos/tan
Beschreibung	Berechnet den Sinus/Cosinus/Tangens des gegebenen Parameters.
Beispiel	sin(90) ergibt 1, cos(0) ergibt 1, tan(45) ergibt 1
Funktion	sqrt
Beschreibung	Berechnet die Quadratwurzel des gegebenen Parameters.
Beispiel	sqrt(16) ergibt 4
Funktion	round, rd
Beschreibung	Rundet Dezimalwerte mathematisch auf angegeben Stellen.
Beispiel	rd(4,65) ergibt 5, rd(4,65;1) ergibt 4,7

Tabelle 4: Einfache numerische Funktionen

Das Verwenden von Funktionen in Formeln mag zu Beginn befremdlich wirken, erfolgt jedoch stets nach dem selben Schema. Mit dem Funktionsnamen wird die gewünschte Funktion angesprochen, mit den Parametern in der nachfolgenden Klammer werden die zu verarbeitenden Daten und Optionen an die Funktion übergeben. Die Funktion liefert anschließend das Ergebnis entweder in eine Variable, oder als Zwischenergebnis zur weiteren Verwendung in einer Formel.

Um den Einstieg zu erleichtern und selten genutzte Funktionen nachzuschlagen, bietet c:geo auf dem Variablen-Reiter, direkt rechts neben dem Variablennamen, eine Schalt-

Funktion	substring, sub
Beschreibung	Extrahiert angegebene Anzahl an Zeichen aus Text.
Beispiel	sub('ABCDE';1;3) ergibt 'BCD'
Funktion	chars, ch
Beschreibung	Extrahiert angegebene Anzahl an Zeichen aus Text.
Beispiel	ch('ABCDE';2;3;4) ergibt 'BCD'
Funktion	length, len
Beschreibung	Berechnet die Länge des Textes (Anzahl der Zeichen).
Beispiel	len('FTF') ergibt 3, len('c geo') ergibt 5

Tabelle 5: Einfache Textfunktionen

fläche mit der Beschriftung „*f*“. Diese öffnet den Dialog „Funktion auswählen“ und zeigt die zur Verfügung stehenden Funktionen an. Durch Auswahl einer der Funktionen kann diese in die jeweilige Variable bzw. Formel übernommen werden. Zu beachten ist, dass bei Auswahl über den Funktionswahl-Dialog stets die Langform des Funktionsnamen verwendet wird. Für Einsteiger mag das sinnvoll sein, wer sich später Tipparbeit sparen möchte, kann auch die jeweilige Kurzform des Funktionsnamens verwenden.

Funktion	icheckSum, ics
Beschreibung	Berechnet die absolute/iterierte Quersumme des angegebenen numerischen Wertes. Berechnet den Buchstabenwert, wenn ein Text angegeben wird.
Beispiel	ics(345) ergibt 3, ics('FTF') ergibt 5
Funktion	checksum, cs
Beschreibung	Berechnet die Quersumme des angegebenen numerischen Wertes. Berechnet den Buchstabenwert, wenn ein Text angegeben wird.
Beispiel	cs(345) ergibt 12, cs('FTF') ergibt 32
Funktion	if
Beschreibung	Wertet Bedingungen aus und gibt davon abhängige Werte zurück.
Beispiel	if(3<4;1;0) ergibt 1, if(3<4;'Wahr';'Falsch') ergibt 'Wahr'

Tabelle 6: Komplexe numerische Funktionen

Die Tabellen 4 bis 7 geben einen Überblick über die zur Verfügung stehenden Funktionen mit deren Langnamen und falls verfügbar auch deren Kurznahmen bzw. Alias. Im Benutzerhandbuch sind die Funktionen unter https://manual.cgeo.org/de/cachevariables#funktionen beschrieben. Derzeit sind einige Funktionen nicht Dokumentiert, bei anderen fehlen die Kurznahmen bzw. Alias. Die im Benutzerhandbuch fehlenden Funktionen und Alias wurden dem c:geo-Quellcode unter https://github.com/cgeo/cgeo entnommen und sind in den Tabellen des Skriptums eingearbeitet.

In Tabelle 4 sind die einfachen numerischen Funktionen zusammengefasst. Sie umfassen das Abschneiden und Runden von Dezimalzahlen, Winkelfunktionen, Berechnen der

Quadratwurzel sowie des Absolutbetrags. Die Kurznahmen der Funktionen „trunc“ und „round“ können ebenfalls verwendet werden.

Tabelle 5 hingegen zeigt einfache Textfunktionen zum Extrahieren von Textteilen oder dem Berechnen der Textlänge bzw. Anzahl der Zeichen eines Textes. Die „length“-Funktion wird im Benutzerhandbuch im Fließtext erwähnt, die Funktionen „substring“ und „chars“ fehlen, können jedoch in c:geo genutzt werden.

Funktion	lettervalue, lv, wordvalue, wv, bww
Beschreibung	Berechnet den Buchstabenwert des angegebenen Textes.
Beispiel	lv('FTF') ergibt 32
Funktion	roman
Beschreibung	Wandelt Text mit römischen Zahlen in den dezimalen Wert um.
Beispiel	roman('VI') erg. 6
Funktion	vanity, vanitycode, vc
Beschreibung	Gibt den Vanity-Code eines Textes zurück.
Beispiel	vanity('cgeo') ergibt 2436
Funktion	rot
Beschreibung	Berechnet einen rotierten Text aus dem angegebenen Text.
Beispiel	rot('abc'; 1) ergibt 'bcd'
Funktion	rot13
Beschreibung	Berechnet den ROT-13 Wert aus dem angegebenen Text.
Beispiel	rot13('abc') ergibt 'nop'

Tabelle 7: Komplexe Textfunktionen

Mit komplexen numerischen Funktionen bietet uns c:geo praktische Werkzeuge, um bei Multi- aber auch (Unknown)Mystery-Caches die Quersumme oder die iterierten Quersumme zu bilden, siehe Tabelle 6. Interessant dabei ist, dass die Funktionen auch Text verarbeiten können. Erneut haben die Funktionen neben ihren Langnamen auch Kurzformen für weniger Tipparbeit. Einen Sonderfall stellt die „if“-Funktion dar, mit der eine oder mehrere Vergleiche oder Bedingungen geprüft werden können.

Schließlich zeigt Tabelle 7 komplexe Textfunktionen wie das Berechnen des Buchstabenwortwerts, das Umrechnen Römischer in Dezimalzahlen, das Anwenden der Verschiebechiffren ROT- und ROT13 sowie das Berechnen des Vanity-Codes[3].

3.7. Kommentare

Durch Nutzung von „#“ können in Formeln Kommentare eingefügt werden. Der Kommentar endet am nächsten „#“ bzw. am Ende der Formel. Alles innerhalb des Kommen-

[3]Eine Methode, Rufnummern durch Buchstaben darzustellen

tars wird bei der Berechnung ignoriert. Der entsprechende Abschnitt im Benutzerhandbuch findet sich unter https://manual.cgeo.org/de/cachevariables#kommentare.

Die Kommentarfunktion kann beispielsweise dazu verwendet werden, um den gesamten Originaltext, den wir von einem Objekt vor Ort abgelesen haben, zu notieren. In die Formel zur Berechnung nehmen wir jedoch nur beispielsweise den ersten Buchstaben auf.

3.8. Überlaufzeichen

In verketteten Ausdrücken, kann das Unterstrich-Zeichen „_“ zur Überlaufkennzeichnung verwendet werden. Es ist ein Platzhalter für mögliche Überläufe wenn eine numerische Variable einen Wert ergibt, der mehr als eine Stelle hat. Ist der Variablenwert einstellig, wird der Platzhalter mit einer 0 befüllt, vgl. https://manual.cgeo.org/de/cachevariables#ueberlaufzeichen.

3.9. Wertebereiche

In Formeln können Wertebereiche durch „[:]“ angegeben werden. Dies kann verwendet werden, um Variablen über einen bestimmten Wertebereich auszuwerten. Ein Beispiel dafür ist die Funktion zur Generierung von Wegpunkten. Details dazu zeigt das Benutzerhandbuch unter https://manual.cgeo.org/de/cachevariables#wertebereiche.

3.10. Fehlerbehebung

Wenn wir mit Variablen arbeiten und diese umbenennen, bleiben ggf. alte Variablennamen in Formeln erhalten. Ebenso kann es vorkommen, dass wir uns vertippt haben und eine Variable fälschlicher Weise angelegt wurde. In diesen Fällen hilft die Schaltfläche „Säubern“. Diese bereinigt und sortiert die Variablen-Liste neu. Die Schaltfläche „Löschen“ hingegen löscht alle Variablen aus der Variablen-Liste.

3.11. Scanne Listing

Die Funktion „Scanne Listing“ untersucht das Geocache-Listing nach potenziellen Formeln und bietet an, diese in den Variablenbereich zu übernehmen, siehe https://manual.cgeo.org/de/cachevariables#kontrollbereich. Da Geocache-Listings für Menschen zum einfachen Lesen gestaltet sind, kann c:geo selten brauchbare Ergebnisse liefern. Es kann

vorkommen, dass Teile einer Formel nicht gelesen und damit abgeschnitten werden. Daher sollte diese Funktion mit großer Vorsicht verwendet werden. Es empfiehlt sich, die Formeln besser per Hand zu kopieren und dabei gleich deren Korrektheit und Vollständigkeit zu prüfen.

4. Persönliche Notiz

Das Eingabefeld für persönlichen Notizen auf https://www.geocaching.com ist ein Textfeld ohne weiterer Funktion. Im Gegensatz dazu weist das Notizfeld in c:geo zusätzliche Funktionen auf, wie unter https://manual.cgeo.org/de/notecoords beschrieben. Tragen wir beispielsweise Koordinaten in das Notizfeld ein, erzeugt c:geo daraus Wegpunkte. Mit Einführung der Cache-Variablen, wie in Abschnitt 3 beschrieben, kam die c:geo-spezifische erweiterte Wegpunkt-Syntax hinzu.

c:geo erzeugt speziellen Code, um Informationen aus Wegpunkten und Variablen im Notizfeld zu sichern. Es ist aber auch möglich – entsprechende Einarbeitung vorausgesetzt, den Code händisch zu erstellen. Beides wollen wir uns in den folgenden Abschnitten näher ansehen.

4.1. Wegpunkte und Variablen in persönlicher Notiz sichern

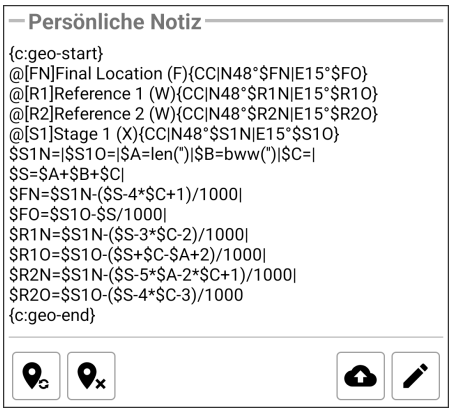


Abbildung 10: Wegpunkte, Variablen und Formeln in persönlicher Notizen übertragen

Nehmen wir beispielsweise einen Multi- oder (Unknown)Mystery-Cache, den wir in c:geo heruntergeladen haben. Für gewöhnlich haben wir mehrere Wegpunkte für die einzelnen Stationen und das Finale angelegt. Eventuell haben wir auch einige Variablen angelegt, zahlreiche Informationen gesammelt und sogar komplexe Berechnungen durchgeführt. In

solchen Fällen möchten wir diese Daten sichern, die Informationen für später aufbewahren oder mit anderen GeocacherInnen teilen.

Das Sichern der eingegebenen Daten erfolgt durch Antippen der Schaltfläche „Synchronisieren“ links unterhalb der persönlichen Notizen, wie unter https://manual.cgeo.org/de/notecoords#wegpunkte_in_der_persoenlichen_notiz_sichern beschrieben. Daraufhin liest c:geo unsere erstellten Wegpunkte, Variablen, Formeln und Kommentare und speichert diese im Notizfeld innerhalb eines speziellen Codeblocks beginnend mit „{c:geo-start}“ und endend mit „{c:geo-end}“, vgl. Abbildung 10.

4.2. Hochladen auf geocaching.com

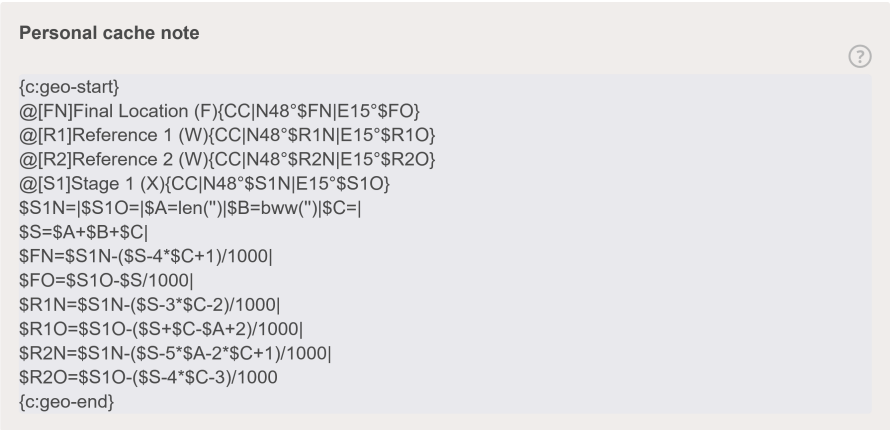


Abbildung 11: Persönliche Notiz hochgeladen auf geocaching.com

Im vorigen Unterabschnitt 4.1 haben wir mit c:geo eine Sicherung unserer Wegpunkte, Variablen und Formeln erstellt. Wenn wir den Geocache aus c:geo löschen, ist nicht sichergestellt, dass diese Sicherung erhalten bleibt. Daher empfiehlt es sich, den Codeblock auf https://www.geocaching.com hochzuladen. Das erledigen wir über die Hochladen-Schaltfläche (zweite von rechts) unterhalb der persönlichen Notiz. Öffnen wir den Geocache nun im Browser auf unserem Personalcomputer, finden wir den Codeblock auch im dortigen Notizfeld, wie Abbildung 11 zeigt.

4.3. Bearbeiten am Computer

Bei sehr langen, aufwändigen Multis mit vielen Wegpunkten oder (Unknown)Mystery-Caches mit komplizierten Berechnungen können wir uns die Arbeit erleichtern und das Erstellen von Wegpunkten, Kopieren von Formeln und Anlegen von Variablen am Personalcomputer zu erledigen. Dazu können wir den gewünschten Codeblock am Com-

puter in einem Texteditor schreiben oder bearbeiten. Die dazu erforderliche Beschreibung der erweiterten Wegpunkt-Syntax findet sich im Benutzerhandbuch unter https://manual.cgeo.org/de/notecoords#erweiterte_wegpunkt-syntax.

Am Beginn der Lernphase können wir uns den Einstieg erleichtern, wenn wir uns Teile des Codes von c:geo erstellen lassen und erst dann eigene Änderungen oder Anpassungen vorzunehmen.

4.4. Herunterladen von geocaching.com

Wir haben nun einen Codeblock mit c:geo erzeugt und auf https://www.geocaching.com hochgeladen oder selbst einen erstellt und im Notizfeld abgelegt. Wenn wir den Geocache nun in c:geo herunterladen oder einen bereits geladenen Geocache aktualisieren, werden die Notizen und damit auch unser Codeblock heruntergeladen. c:geo verarbeitet daraufhin den Codeblock und erstellt bzw. aktualisiert Wegpunkte, Variablen, Formeln und Kommentare.

Standardmäßig ist c:geo so eingestellt, dass Notizen nicht überschrieben werden. Ändern wir beispielsweise unsere Notiz auf https://www.geocaching.com und aktualisieren den Geocache in c:geo, wird die geänderte Version zusätzlich ins Notizfeld geladen. Wer das nicht möchte, kann das Verhalten in den Einstellungen zu den „Cache-Details“ ändern, siehe https://manual.cgeo.org/de/mainmenu/settings#cache-details, „Persönliche Cache-Notiz überschreiben“.

4.5. Fehlerbehebung

Wenn wir häufig Änderungen an Wegpunkten, Variablen oder insbesondere dem Codeblock vornehmen, kommt es vor, dass alte bzw. falsche Wegpunkte erhalten bleiben. Auch in der Variablenliste bleiben alte bzw. falsche erhalten oder es kommt mitunter die Sortierreihenfolge abhanden.

Vorausgesetzt, die Informationen im Codeblock des Notizfeldes sind auf letztgültigem Stand, kann man über das „Drei-Pünktchen-Menü“ rechts oben im Bereich „Wegpunkte“ die „Benutzerdefinierten Wegpunkte löschen“. c:geo erstellt dann nur mehr jene Wegpunkte, die im Codeblock vorhanden sind.

Ebenso kann man mit den Variablen verfahren. Die Schaltfläche „Säubern“ bereinigt und sortiert die Variablen, „Löschen“ löscht alle Variablen (nur auf der Seite, nicht im Notizfeld).

5. Multi- und (Unknown)Mystery-Caches fehlerfrei

Wir absolvieren einen schönen Multi- oder spannenden (Unknown)Mystery-Cache mit mehreren Stages. Erfahrungsgemäß passiert es früher oder später, dass sich irgendwo ein Fehler einschleicht, wir ins Straucheln kommen oder gar erfolglos abbrechen müssen. Der Fehler kann beim zu hastigen Lesen des Geocache-Listings, beim Übertragen der Zahlen und Formeln auf den Notizblock, beim Eintippen in den Taschenrechner oder die Taschenrechner-App am Smartphone, beim Ablesen des Rechenergebnisses oder beim Eintippen der Koordinaten für den neuen Wegpunkt passieren. Ebenso können wir es mit komplizierten oder aufwändigen Berechnungen zu tun bekommen, die bei ungünstigen Witterungsbedingungen wie Hitze, Kälte oder Nässe unangenehm und mühsam werden, unsere Konzentration schwinden lassen und weitere Fehlerquellen öffnen.

Wie bereits in Abschnitt 3 erwähnt, sind mit c:geo ab Version 2022.06.06 der überarbeitete Wegpunkt- bzw. Koordinaten-Rechner und die Variablen-Registerkarte eingeführt worden. Ebenso wurden Variablen für Wegpunktberechnungen global pro Geocache nutzbar. Das bietet uns nun die Möglichkeit, mit Wegpunkten bei Multi- und (Unknown)Mystery-Caches zu rechnen, obwohl wir deren Koordinaten noch nicht kennen. Wir ersetzen also geschickt Koordinaten durch Variablen.

Bringen wir all die Stolpersteine, Fehlerquellen und Mühsale mit den neuen Funktionen in c:geo zusammen, so können wir uns künftig auf Multi- und (Unknown)Mystery-Caches bereits vorab vorbereiten. Zahlreiche Aufgaben wie das Kopieren und Kontrollieren von Formeln, das Übertragen von Koordinaten sowie das Anlegen von Wegpunkten können wir nun zu Hause, im warmen bzw. kühlen und trockenen Wohn- oder Arbeitszimmer in Ruhe und ohne Zeitdruck erledigen. Dadurch ersparen wir uns zahlreiche typische Fehler, die aufwändige Berechnung übernimmt für uns c:geo und wir müssen uns nur um das Ablesen der Informationen auf Objekten vor Ort und das Befüllen der Variablen kümmern. Dabei wird jeder Multi- oder (Unknown)Mystery-Cache zum Genuss und wir können beim Geocachen vollends die Landschaft genießen.

Dazu ist natürlich etwas Vorbereitung und Planung nötig. Die wichtigsten Punkte wollen wir uns in den folgenden Abschnitten genauer ansehen.

5.1. Vorbereiten

Bei der Vorbereitung vorab zu Hause oder einer vorübergehenden Homebase muss das Geocache-Listing gelesen und dahingehend untersucht werden, ob eine Vorbereitung überhaupt sinnvoll ist. Bei Geocaches, wo sich die Koordinaten für die jeweils nächste Station erst vor Ort finden, können wir kaum etwas vorbereiten.

Haben wir beispielsweise einen AdventureLab® Bonus-Cache oder einen Multi-Cache wie „Schulstadt Horn“ (https://coord.info/GC5PRRP), können wir ebenso wenig vorbereiten. Wir können einen Wegpunkt für das Finale anlegen und im geführten Modus des Wegpunkt-Rechners die Ziffern durch die Variablen ersetzen. In Listing 1 sehen wir den Aufbau eines entsprechenden Codeblocks.

```
{c:geo-start}
@Finale (F) {CC|N48°40.ABC'|E15°39.DEF'|DMM}
$A=cs() | $B= | $C= | $D=len('') | $E= | $F=
{c:geo-end}
```

Listing 1: Codeblock für „Schulstadt Horn“ GC5PRRP

Etwas mehr vorbereiten können wir, wenn wir einen Multi-Cache vor uns haben, bei dem sich die Stationen unabhängig voneinander berechnen. Ein gutes Beispiel dafür ist „Im Wald ist Ruh vs. (Wald-)Autobahn“ (https://coord.info/GCA00Y6). Der Cache-Owner liefert uns sogar eine Final-Formel, die wir eins-zu-eins in den Wegpunkt-Rechner kopieren können. Außerdem können wir in c:geo einen „Mini-GeoChecker“ nachbauen, indem wir mit der „ics“-Funktion die iterierte Ziffernsumme der Variable „S“ berechnen und mit der „if“-Funktion den Vergleich mit der Prüfsumme anstellen. Ist das Ergebnis des Vergleichs wahr, gibt c:geo „Richtig“ aus, andernfalls „Falsch“. Listing 2 zeigt einen entsprechend aufgebauten Codeblock.

```
{c:geo-start}
@Station 2 (S) {CC|N48°17.(703-A)|E15°31.(803-A)}
@Finale (F) {CC|N48°(15+(S+610)/1000)'|E15°(28+(2*S-80)/1000)'}
$A= | $B= | $C= | $S=A+B+C |
$CHKSUM=if(ics(S)==9;'Richtig';'Falsch')
{c:geo-end}
```

Listing 2: Codeblock für „Im Wald ist Ruh vs. (Wald-)Autobahn“ GCA00Y6

Sehr gut vorbereiten können wir einen Multi-Caches, bei dem ausgehend von der ersten Station zur nächsten Station und bis zum Finale gerechnet wird. Ein gutes Beispiel hierfür ist „Slawischer Fürst *reloaded*“ (https://coord.info/GC77E0R). Hier nutzen wir die globalen Cache-Variablen und legen parallel zu den Wegpunkten entsprechende Variablen für die noch unbekannten Stationen an.

Die erste Station ist durch die gelisteten Koordinaten bzw. durch einen Startpunkt gegeben. Wir legen neue Variablen für diese erste Station an, benennen sie beispielsweise „S1N“ und „S1E“ und tragen die Koordinatenwerte ein. Für die zweite Station legen wir ebenso Variablen an und benennen diese beispielsweise „S2N“ und „S2E“. Auf die Koordinatenwerte der erste Station können wir nun über die Variablen „$S1N“ und „$S1E“ zugreifen. Wir kopieren uns die Formeln für Station 2 aus dem Geocache-Listing, ersetzen die Koordinaten von Stage 1 durch unsere Stationsvariablen „$S1N“ und „$S1E“ und können so die (Vor-)Berechnung der zweiten Station fertigstellen.

Wahrscheinlich wird uns c:geo nun eine Fehlermeldung bringen, weil wir für die Variablen

aus den Formeln des Geocache-Listings noch keine Werte haben. Diese Fehler ignorieren wir fürs Erste. Wichtig ist zu erkennen, dass wir die zweite Station unseres Multis bereits bestimmt haben, obwohl wir deren konkreten Koordinatenwerte noch gar nicht kennen. Trotzdem können wir die Stationsvariablen bereits zum Berechnen der nächsten Wegpunkte sowie des Finals verwenden. Wir legen also für jede weitere Station entsprechende Variablen an. In den Formeln aus dem Geocache-Listing ersetzten wir stets die Koordinaten der vorherigen Station durch unsere Stationsvariablen. So verfahren wir auch mit den Variablen für das Finale, die wir beispielsweise „$FN“ und „$FE“ nennen. Wir ersetzen auch hier in der Finalformel die Koordinatenwerte der vorherigen Station durch unsere Stationsvariablen „$S7N“ und „$S7E“.

Abschließend können wir die zugehörigen Wegpunkte erstellen. Im Wegpunkt-Rechner geben wir bei Station 2 die Variablen „$S2N“ und „$S2E“ an, bei Station 3 „$S3N“ und „$S3E“ u.s.w. Im Wegpunkt zum Finale sind es die Variablen „$FN“ und „$FE“. Zusammengefasst ergibt das einen Codeblock wie Listing 3 zeigt.

```
{c:geo-start}
@Stage 2 (X) {CC|N48°$S2N|E15°$S2E}
@Stage 3 (X) {CC|N48°$S3N|E15°$S3E}
@Stage 4 (X) {CC|N48°$S4N|E15°$S4E}
@Stage 5 (X) {CC|N48°$S5N|E15°$S5E}
@Stage 6 (X) {CC|N48°$S6N|E15°$S6E}
@Stage 7 (X) {CC|N48°$S7N|E15°$S7E}
@Final (F) {CC|N48°$FN|E15°$FE}
$A= | $B= | $C= | $D= | $E= | $F= | $G= | $H= | $I= | $K=cs() | $M=lv('') | $N= | $P= |
$S1N=35.352 |
$S1E=37.936 |
$S2N=$S1N+(-$A*$B-$B-10)/1000 |
$S2E=$S1E+($A*15-5)/1000 |
$S3N=$S2N+(-2*$C+2)/1000 |
$S3E=$S2E+(3*$D+1)/1000 |
$S4N=$S3N+(8*$E-(2*$D-1))/1000 |
$S4E=$S3E+(15*$F+2)/1000 |
$S5N=$S4N+($H*$C-$B)/1000 |
$S5E=$S4E+($G+5*$C-2)/1000 |
$S6N=$S5N+(-$I*$I)/1000 |
$S6E=$S5E+(-3*$D)/1000 |
$S7N=$S6N+(2*$K-2)/1000 |
$S7E=$S6E+(-$M*$C+97)/1000 |
$FN=$S7N+($F+$N+$P)/1000 |
$FE=$S7E+(-$G-$N*$C-2*$P-99)/1000
{c:geo-end}
```

Listing 3: Codeblock für „Slawischer Fürst *reloaded*“ GC77E0R

Damit haben wir den Multi-Cache vollständig vorab berechnet. Alle Variablen sind angelegt und alle Wegpunkte vordefiniert. Sobald wir die Variablen mit den Werten befüllen, die wir auf den Objekten vor Ort abgelesen haben, berechnet c:geo die Stationsvariablen. Darüber erhalten schließlich unsere vordefinierten Wegpunkte konkrete Koordinaten. Mit jeder gesammelten Information berechnet c:geo den nächsten Wegpunkt bis wir uns schließlich beim korrekt berechneten Finale im Logbuch eintragen können.

5.2. Variablennamen planen

Bei Multi-Caches mit mehr als 10 Stationen sollten wir uns Gedanken über die Benennung der Stationsvariablen machen. c:geo sortiert in ASCII-Reihenfolge, was uns am Reiter der Variablen zur falschen Reihung „S1N“, „S10N“, „S2N“ führt. Um hier die Ordnung wiederherzustellen, können wir führende Nullen im Variablennamen verwenden. Damit sortiert c:geo die Variablen-Liste wie erwartet: „S01N“, „S02N“, „S10N“.

Ein Beispiel für einen Geocache, bei dem man sich jedenfalls Gedanken über die Benennung der (Stations-)Variablen machen muss, ist „Welterbesteig Wachau“ (https://coord.info/GC8PWHQ). Mit 102 Variablen und 14 Prüfsummenvariablen ist eine geordnete Variablen-Liste unumgänglich um den Überblick zu behalten. Folglich könnten wir uns überlegen, als Namen für die Variablen „V01A“, „V01B“, „V01C“, ... „V02A“, „V02B“, „V02C“, für die Prüfsummen „V01X“, „V02X“, „V03X“, ... und für iterierten Ziffernsummen „ZSA“, „ZSB“, „ZSC“, ... zu verwenden.

Ein entsprechend ausgearbeiteter Codeblock zum „Welterbesteig Wachau“ findet sich – aufgrund der Länge – neben weiteren Beispielen im Listing 17 in Anhang A.

5.3. Koordinatenformat nach Cache-Gebiet wählen

Führt ein Geocache über einen Konfluenzpunkt[4] bzw. über einen ganzzahligen Längen- oder Breitengrad, so muss man beim Rechnen im „DDD°MM.MMM′“-Format einen möglichen Übertrag auf die Grade beachten. Im Kleinen zeigt sich die Problematik auch, wenn die Dezimalminuten kleiner als Null oder größer als Tausend werden, da man mit einem Übertrag auf die Minuten rechnen muss.

In beiden Fällen können wir uns überlegen, statt dessen im „DDD.DDDDD°“-Format zu rechnen. Dazu wandeln wir die Startkoordinaten wie folgt um: Grad + Minuten / 60. Ein Großteil der Formeln in Geocache-Listings ist so gestaltet, dass man in den Milliminuten rechnet, den Dezimalpunkt wegdenkt oder die Dezimalminuten gedanklich durch 1000 dividiert. Aus das berücksichtigen wir und dividieren den geklammerten Formelausdruck durch 60000 (60 für die Minuten und 1000 für die Dezimalminuten).

Weil dieses Vorgehen bei einem Großteil der Multi-Caches funktioniert, können wir uns eine Codeblock-Vorlage erstellen. Diese ist in Listing 4 abgebildet. Die drei Punkte „...“ dienen als Platzhalter für die Formeln aus dem Geocache-Listing. Auf Basis dieser Vorlage können wir uns schnell einen passenden Codeblock für viele Multi-Caches erstellen.

Der genaue Leser wird erkannt haben, dass in der Codeblock-Vorlage bei den Wegpunk-

[4] Schnittpunkt eines ganzzahligen Längengrades mit einem ganzzahligen Breitengrad

ten die Richtungsangabe „N“ bzw. „S“ und „E“ bzw. „W“ fehlen. Beim Rechnen im „DDD.DDDDD°“-Format müssen wir diese nicht zwingend angeben, c:geo erkennt die Richtung anhand des Vorzeichens: „N“ bzw. „E“ bei positivem Vorzeichen, „S“ bzw. „W“ bei negativem Vorzeichen. In den besonders außergewöhnlichen Gegenden auf dem Äquator oder dem Nullmeridian sollte der Berechnung und dem Ergebnis ohnehin besondere Beachtung geschenkt werden.

```
{c:geo-start}
@[P1]Parkplatz (P) {CC|$S1N°|$S1E°}
@[S2]Station 2 (S) {CC|$S2N°|$S2E°}
@[F1]Finale (F) {CC|$FN°|$FE°}
$S1N=48+40.340/60 |
$S1E=15+39.256/60 |
$S2N=$S1N+( ... )/60000 |
$S2E=$S1E+( ... )/60000 |
$FN=$S2N+( ... )/60000 |
$FE=$S2E+( ... )/60000
{c:geo-end}
```

Listing 4: Codeblock-Vorlage zum Rechnen im „DDD.DDDDD°“-Format

6. Beispiele, Nützliches und Kurioses

In diesem Abschnitt wollen wir das bisher Gelernte anhand einiger praktischer Beispiele vertiefen. Dabei werden wir uns zwischen Nützlichem und Kuriosem bewegen, stets mit dem Fokus auf konkreten Lösungen aus dem und für den Geocaching-Alltag.

6.1. Prüfsumme als Geochecker-Ersatz

In Geocache-Listings finden wir unterschiedliche Geochecker, um unsere berechneten Koordinaten zu überprüfen. Manchmal finden wir jedoch eine (iterierte) Prüfsumme über beispielsweise alle gefundenen Variablenwerte. Das haben wir bereits in Abschnitt 5 beim Geocache „Im Wald ist Ruh vs. (Wald-)Autobahn“ (https://coord.info/GCA00Y6) gesehen. Dabei nutzen wir die „if“-Funktion aus Tabelle 6 um eine Prüfung mit dem Ergebnis „Richtig“ oder „Falsch“ zu erstellen.

c:geo kann aber auch Emoticons bzw. Unicode-Zeichen verarbeiten und anzeigen. Das können wir nutzen, um die Richtigkeit einer Prüfsumme mit grünem Häkchen / rotem Kreuzchen oder etwa auch mit lachenden und weinenden Smiley-Gesichtern anzuzeigen. Abbildung 12 zeigt Screenshots, wie solche Checks auch gestaltet werden können. Eingegeben werden diese Zeichen über die virtuelle Tastatur am Smartphone.

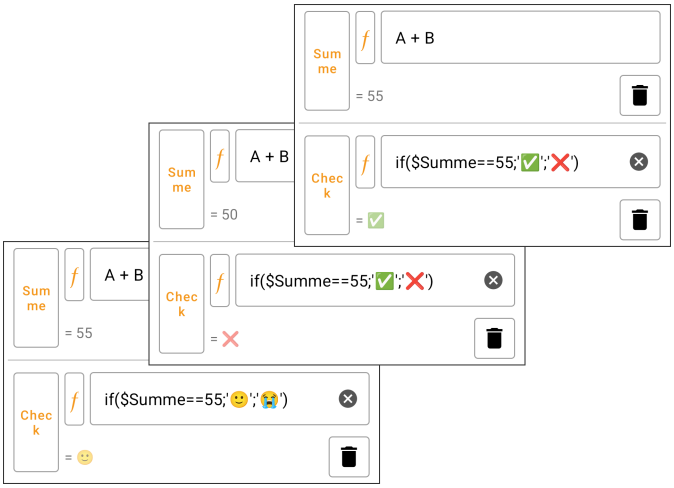


Abbildung 12: Klare Anzeige des Ergebnisses durch Emoticons

6.2. Ungewöhnliche Formeln und Koordinatenformate

Geocache-Listings können auch Rechenanweisungen und resultierende Koordinaten in unüblicher Form enthalten. Mit etwas Kreativität können wir auch in diesen Fällen mit c:geo rechnen. Sehen wir uns als Beispiel den (Unknown)Mystery-Cache „#13 Mödring Runde - Sommerfrische“ (https://coord.info/GCAC6TD) an. Sofort fällt auf, dass die Formeln mehrere Worte lange Ausdrücke enthalten. Da c:geo in Variablennamen keine Leerzeichen erlaubt, müssen wir uns eigene Variablennamen ausdenken. Damit wir die Übersicht behalten, fügen wir die ursprünglichen Ausdrücke aus dem Geocache-Listing als Kommentare neben den neuen Variablennamen hinzu. Nun ersetzen wir noch die Ausdrücke in den Formeln durch unsere eigenen Variablen und können den Rechenaufwand an c:geo abgeben. Listing 5 zeigt eine mögliche Lösung.

```
{c:geo-start}
@[FN]Final Location (F) {CC|N$ND°$NM|E$OD°$OM}
$A= # Zwettler Bad an jetzigem Standort |
$B= # Eröffnung der Kamptalbahn |
$C= # Seehöhe Bahnhof Kamegg |
$D= # Preis für Schwimmbekleidung Frauen |
$E= # Beginn der Schwimmschule |
$F= # Saisonkarte Kinder |
$G= # Schwimmbekleidung für Männer |
$N=$A^2+$B*$C*2.3+$C*$D*1.93-6 |
$ND=sub($N;0;2) # Nord Grade |
$NM=sub($N;2;5)/1000 # Nord Dezimalminuten |
$O=$E^2/$F-$B*$G^2*1.15+$C*3.6+$G-2 |
$OD=sub($O;0;2) # Ost Grade |
$OM=sub($O;2;5)/1000 # Ost Dezimalminuten
{c:geo-end}
```

Listing 5: Codeblock für „#13 Mödring Runde - Sommerfrische“ GCAC6TD

Liegen uns nun die Ergebnisse für die Nord- und Ostkoordinaten vor, müssen wir feststellen, dass diese in keinem – von c:geo direkt – verarbeitbaren Format vorliegen. Wir finden die Grade, und Dezimalminuten zusammengeklebt als „DDMMMMM“ vor. Hier müssen wir c:geo etwas unterstützen, indem wir die Grade und Minutenanteile mit der „sub“-Funtion trennen und die Minuten in Dezimalminuten umrechnen. Zusammengefasst ergibt das eine Lösung wie beispielsweise in Listing 5 gezeigt.

6.3. Tabellarische Verarbeitung, generierte Hinweise

In Geocache-Listings können wir auch Berechnungen in tabellarischer Form und weitere textuelle Auswertungen finden. Am Beispiel des Geocache „V#4 - Vogelkunde“ (https://coord.info/GCAM739) wollen wir uns das näher ansehen. Unter dem Rätselbild fällt uns sofort die Auswertungstabelle auf. Darin werden die Lösungsworte eingetragen und Buchstabenwortwerte sowie iterierte Quersummen berechnet. Außerdem gibt es markierte Felder, aus denen sich ein zusätzlicher Hinweis zum Versteckort ergibt.

Das sieht auf den ersten Blick komplex aus, doch auch hier kann uns c:geo Rechenarbeit abnehmen. Wir legen gemäß der linken Spalte die Variablen an, als Werte geben wir die Lösungsworte unter Anführungszeichen ein. Damit erhalten wir uns ausreichend Flexibilität für die späteren Berechnungen und Auswertungen. Danach legen wir uns weitere Variablen gemäß den rechten beiden Spalten an. Aus den einzelnen Lösungsworten berechnen wir die Buchstabenwortwerte mit der Funktion „lettervalue“, kurz „lv“ sowie die iterierten Quersummen mit der Funktion „icheckSum“, kurz „ics“. Diese Variablen fließen dann in die Formel für die Finalkoordinaten ein.

```
{c:geo-start}
@[FN]Final (F) {CC|$SN2|$SO2}
$A='' | $B='' | $C='' | $D='' | $E='' | $F='' | $G='' | $H='' | $I='' | $J='' | $K='' | $L='' |
$M='' | $N='' | $O='' | $P='' | $Q='' | $R='' |
$SHINT=$SHW1' '$SHW2' '$SHW3 |
$SHW1=ch($A;len($A)-1)ch($D;len($D)-4)ch($E;len($E)-4)ch($F;len($F)-9) |
$SHW2=ch($I;len($I)-10;len($I)-7)ch($K;len($K)-8)ch($L;len($L)-7)ch($M;len($M)-7) |
$SHW3=ch($N;len($N)-6)ch($O;len($O)-4;len($O)-3)ch($Q;len($Q)-10)ch($R;len($R)-3) |
$SN1=48+21/60 |
$SO1=15+46/60 |
$SN2=$SN1+(-($WA+$WB+$WC+$WD+$WE+$WF+$WG+$WH+$WI+$WP)+$WJ+$WR)/60000 |
$SO2=$SO1+($WA+$WB+$WC+$WD+$WE+$WF+$WG+$WH+$WI+$WK+$WL+$WM+$WN*$WO-$WJ-$WO-$WO)/60000 |
$WA=lv($A) | $WB=lv($B) | $WC=lv($C) | $WD=lv($D) | $WE=lv($E) | $WF=lv($F) | $WG=lv($G) |
$WH=lv($H) | $WI=lv($I) | $WJ=ics(lv($J)) | $WK=lv($K) | $WL=lv($L) | $WM=lv($M) | $WN=lv($N) |
$WO=ics(lv($O)) | $WP=lv($P) | $WQ=lv($Q) | $WR=ics(lv($R))
{c:geo-end}
```

Listing 6: Codeblock für „V#4 - Vogelkunde“ GCAM739

Da wir zuvor die Lösungsworte als Text in Variablen gespeichert haben, können wir noch den Hinweis mit c:geo auswerten. Hier nutzen wir die Funktion „chars“, kurz „ch“ um uns einzelne Buchstaben aus den Lösungsworten herauszupicken. Über den Verkettungsmechanismus bauen wir uns schließlich den Hinweis zusammen. Zusammengefasst ergibt

das den Codeblock wie in Listing 6 gezeigt. Aufgrund der Formellängen und zur besseren Lesbarkeit werden die einzelnen Worte des Hinweises einzeln ausgewertet. Natürlich könnte man die Verkettung auch in einer einzelnen Codezeile zusammenfassen.

6.4. Multi-Cache mathematisch abkürzen

Bei dem Multi-Cache „HORN iST VORN - Satz des Pythagoras“ (https://coord.info/GC5F4AG) steckt die Besonderheit bereits im Titel. Der Owner erlaubt uns, mit dem Satz des Pythagoras[5] eine Abkürzung zu nehmen. Das wollen wir aufgreifen und führen diese Berechnung mit c:geo und der Wurzelfunktion „sqrt“ durch. In Listing 7 sind die Rechenschritte dargestellt.

```
{c:geo-start}
$A= |
$B=sqrt(C^2-A^2) |
$C=
{c:geo-end}
```

Listing 7: Codeblock für „HORN iST VORN - Satz des Pythagoras“ GC5F4AG

Für Stage 2 bietet es sich an, einen Wegpunkt mit den gelisteten Koordinaten anzulegen und mit der berechneten Distanz und dem Winkel aus dem Listing eine Wegpunkt-Projektion wie in Unterabschnitt 2.2 gezeigt durchzuführen.

6.5. Entfernung zwischen zwei Punkten auf der „Erd-Kugel“

Weil uns c:geo auch trigonometrische Funktionen („Winkelfunktionen“) zur Verfügung stellt, können wir damit durchaus komplexe Berechnungen anstellen. Einen darauf zugeschnittenen Geocache finden wir mit „Recherchieren und Rechnen“ (https://coord.info/GCA5R36). Bei der dritten Frage ist die Entfernung zwischen zwei Koordinatenpunkte gesucht. Nach etwas Vorüberlegung anhand eines Näherungsverfahrens[6] ist das mit wenigen Rechenschritten[7] erledigt. Dazu etwas Theorie:

Gegeben seien ein Punkt $A(\phi_1, \lambda_1)$ durch seinen Breitengrad ϕ_1 und Längengrad λ_1 sowie ein Punkt $B(\phi_2, \lambda_2)$ durch seinen Breitengrad ϕ_2 und Längengrad λ_2. Zu bestimmen ist die Distanz d zwischen den beiden Punkten.

[5] https://de.wikipedia.org/wiki/Satz_des_Pythagoras
[6] https://de.wikipedia.org/wiki/Wegpunkt-Projektion
[7] https://www.kompf.de/gps/distcalc.html

$$
\begin{aligned}
R &\approx 6371{,}003989\,\text{km} \\
d &= \frac{\overline{AB}}{R} \cdot \frac{180^\circ}{\pi} = \frac{\overline{AB}}{\frac{6371\text{km}\cdot\pi}{180^\circ}} = \frac{\overline{AB}}{111{,}195\,\text{km}} \quad \text{(in Grad)} \\
\Delta_x &= 111{,}2 \cdot \cos\left(\frac{\phi_1 + \phi_2}{2}\right) \cdot (\lambda_1 - \lambda_2) \\
\Delta_y &= 111.2 \cdot (\phi_1 - \phi_2) \\
d &= \sqrt{\Delta_x^2 + \Delta_y^2}
\end{aligned}
$$

Wir haben uns damit eine Näherungsformel zur Berechnung der Distanz zwischen zwei Koordinatenpunkten auf der „Erd-Kugel“ erarbeitet. Nun können wir konkrete Punkte einsetzen und beispielsweise die Entfernung zwischen dem MEGA-Event und dem Rathaus der Niederösterreichischen Landeshaupt St. Pölten berechnen:

$$
\begin{aligned}
lat_1 &= 48 + \frac{40{,}340}{60} = 48{,}672333 \quad \text{(in Grad)} \\
lon_1 &= 15 + \frac{39{,}256}{60} = 15{,}654267 \quad \text{(in Grad)} \\
lat_2 &= 48 + \frac{12{,}265}{60} = 48{,}204417 \quad \text{(in Grad)} \\
lon_2 &= 15 + \frac{37{,}375}{60} = 15{,}622917 \quad \text{(in Grad)} \\
d_x &= 111{,}2 \cdot \cos\left(\frac{48{,}672333 + 48{,}204417}{2}\right) \cdot (15{,}654267 - 15{,}622917) \\
&= 2{,}31278\,\text{km} \\
d_y &= 111.2 \cdot (48{,}672333 - 48{,}204417) = 52{,}032333\,\text{km} \\
d &= \sqrt{2{,}31278^2 + 52{,}032333^2} = 52{,}083708\,\text{km}
\end{aligned}
$$

Wir erhalten als Ergebnis eine Entfernung von rund 52km. Das Ergebnis berücksichtigt die Erdkrümmung und versteht sich als kürzeste Entfernung (Luftlinie). Das Näherungsverfahren ist für einen Großteil der Berechnungen beim Geocaching ausreichend. In einen Codeblock übertragen stellt sich die Berechnung wie in Listing 8 dar.

```
{c:geo-start}
$lat1=48+40.340/60 |
$lon1=15+39.256/60 |
$lat2=48+12.265/60 |
$lon2=15+37.375/60 |
$d=sqrt((111.2*cos(($lat1+$lat2)/2)*($lon1-$lon2))^2+(111.2*($lat1-$lat2))^2)
{c:geo-end}
```

Listing 8: Codeblock zur Entfernung zwischen zwei Punkten

Auf den ursprünglichen Geocache „Recherchieren und Rechnen“ (https://coord.info/GCA5R36) angewendet, berechnen wir die Distanz zwischen den beiden gesuchten Koordinatenpunkten mit den zuvor erarbeiteten Formeln. Das Ergebnis speichern wir in der Variablen „$distance“. Da für Variable „C“ die gerundete Entfernung gesucht ist, verwenden wir die Funktion „round“, kurz „rd“ um den Wert der Variablen „$distance“ auf den ganzzahligen Wert zu runden. Im Listing 9 ist der resultierende Codeblock abgebildet.

```
{c:geo-start}
@Finale (F) {CC|N48°10.$XXX'|E015°37.$YYY'}
$A=bww('')*cs() |
$B=bww(rot13(''))*(...+...) |
$C=round($distance) |
$D=ics(A+B) |
$F=cs()*... |
$G=len('') |
$H=...*vanity('') |
$lat1=..+...../60 |
$lon1=..+...../60 |
$lat2=..+...../60 |
$lon2=..+...../60 |
$dx=111.2*cos(($lat1+$lat2)/2)*($lon1-$lon2) |
$dy=111.2*($lat1-$lat2) |
$distance=sqrt($dx^2+$dy^2) |
$XXX=A-C-4*(B+D) |
$YYY=H-12*F-21*G-10
{c:geo-end}
```

Listing 9: Codeblock für „Recherchieren und Rechnen“ GCA5R36

Um nicht zu viele Geheimnisse des Rätsels vorweg zu verraten, wurden die gesuchten Information im Listing durch Punkte „...“ ersetzt. Die Informationen müssen selbst recherchiert und dann eingegeben werden. Abschließend sei noch angemerkt, dass dieses Rätsel einige weitere besondere Formeln, wie beispielsweise die Wortlänge mit der Funktion „length“, kurz „len“ oder auch dem Vanity-Code mit der Funktion „vanity“, kurz „vc“ erfordert.

6.6. Wegpunkt-Projektion

In heiterer Runde beim „4. St. Pöltner Geocache Stammtisch“ (https://coord.info/GCA46TT) ist die Idee entstanden, Geocaches aus der Kategorie „Waypoint Challenge #1“ (https://coord.info/GC3N56E) mittels c:geo zu berechnen. Die einzelnen Stationen berechnen sich dabei stets mittels Wegpunkt-Projektion. „Challenge accepted“! Nach etwas Vorüberlegung anhand eines Näherungsverfahrens[8] und mit wenigen Rechenschritten[9] ist auch das möglich. Dazu wieder etwas Theorie:

[8] https://de.wikipedia.org/wiki/Wegpunkt-Projektion
[9] https://www.cachewiki.de/wiki/Wegpunktprojektion

Gegeben seien ein Punkt $A(\phi_1, \lambda_1)$ durch seinen Breitengrad ϕ_1 und Längengrad λ_1, ein Peilwinkel α und eine Distanz $\overline{AB}$ zu einem unbekannten Punkt $B(\phi_2, \lambda_2)$. Zu bestimmen sind der Breitengrad ϕ_2 und der Längengrad λ_2 des Zielpunktes.

$$
\begin{aligned}
R_{Erde} &\approx 6371004\,\mathrm{m} \\
d &= \frac{\overline{AB}}{R_{Erde}} \cdot \frac{180° \cdot 60'}{\pi} = \frac{\overline{AB}}{\frac{6371004\mathrm{m} \cdot \pi}{180° \cdot 60'}} = \frac{\overline{AB}}{1853{,}25\,\mathrm{m}} \quad \text{(in Bogenminuten)} \\
\phi_2 &= \phi_1 + \frac{\frac{\overline{AB}}{1853} \cdot \cos(\alpha)}{60} \\
\lambda_2 &= \lambda_1 + \frac{\frac{\overline{AB} \cdot \sin(\alpha)}{1853 \cdot \cos(\phi_2)}}{60}
\end{aligned}
$$

Wir haben uns damit eine Näherungsformel zur Berechnung der Wegpunkt-Projektion erarbeitet. Mit den Projektionsdaten, Entfernung und Winkel, können wir eine konkrete Wegpunkt-Projektion beispielsweise vom MEGA-Event zum 590 Meter entfernten und im Winkel von 92.7° verorteten CITO am Freigelände durchführen.

$$
\begin{aligned}
dist &= 590 \\
alpha &= 92{,}7 \\
lat_1 &= 48 + \frac{40{,}340}{60} = 48{,}672333 \quad \text{(in Grad)} \\
lon_1 &= 15 + \frac{39{,}256}{60} = 15{,}654267 \quad \text{(in Grad)} \\
lat_2 &= lat_1 + \frac{\frac{590}{1853} \cdot \cos(92{,}7)}{60} = 48{,}672083 \\
lon_2 &= lon_1 + \frac{\frac{590 \cdot \sin(92{,}7)}{1853 \cdot \cos(48{,}672083)}}{60} = 15{,}662294
\end{aligned}
$$

```
{c:geo-start}
@Projektion (W) {CC|$lat2|$lon2}
$dist=590 |
$alpha=92.7 |
$lat1=48+40.340/60 |
$lon1=15+39.256/60 |
$lat2=$lat1+($dist/1853*cos($alpha))/60 |
$lon2=$lon1+(($dist*sin($alpha))/(1853*cos($lat2)))/60 |
{c:geo-end}
```

Listing 10: Codeblock zur Wegpunkt-Projektion

Wir erhalten als Ergebnis die Nord- und Ostkoordinaten des projizierten Koordinatenpunkts. Das Ergebnis berücksichtigt die Erdkrümmung, damit ist dieses Näherungsverfahren für einen Großteil der Berechnungen beim Geocaching ausreichend. In einen Codeblock übertragen stellt sich die Berechnung wie in Listing 10 dar.

Zurück zur ursprünglichen Idee zur Vorabberechnung der „Waypoint Challenge #1“ (https://coord.info/GC3N56E) mittels c:geo: Wir legen uns Variablen A bis P und berechnen damit pro Station Distanz und Winkel, die wir ebenfalls in Variablen speichern. Diese Variablen verwenden wir zur Berechnung der Stations-Variablen. Abschließend erstellen wir noch die Wegpunkte basierend auf den Stations-Variablen. Schon sind wir bestens vorbereitet und können die „Waypoint Challenge #1“ bestreiten.

Aufgrund der Länge des Codeblocks zur „Waypoint Challenge #1“ ist dieser, neben weiteren Beispielen, im Listing 15 in Anhang A zu finden.

6.7. Weitere Beispiele

Zum Abschluss sei noch auf Anhang A mit weiteren Codebeispielen verwiesen. Diese können weitestgehend eins-zu-eins kopiert und in c:geo verwendet werden. Eine Warnung vorweg: Geocache-Listings können sich ändern und Geocaches können verlegt werden. Daher bitte alle Codeblöcke und Formeln VOR dem Outdoor-Einsatz prüfen!

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

A. Anhang

```
{c:geo-start}
@[P1]Parkplatz (P) {CC|N48°$S1N'|E15°$S1E'}
@[S2]Station 2 (S) {CC|N48°$S2N'|E15°$S2E'}
@[F1]Finale (F) {CC|N48°$FN'|E15°$FE'}
$S1N=40.340 |
$S1E=39.256 |
$S2N=$S1N+( ... )/1000 |
$S2E=$S1E+( ... )/1000 |
$FN=$S2N+( ... )/1000 |
$FE=$S2E+( ... )/1000
{c:geo-end}
```

Listing 11: Codeblock-Vorlage zum Rechnen im „DDD°MM.MMM-Format“

```
{c:geo-start}
@[P1]Parkplatz (P) {CC|$S1N°|$S1E°}
@[S2]Station 2 (S) {CC|$S2N°|$S2E°}
@[F1]Finale (F) {CC|$FN°|$FE°}
$S1N=48+40.340/60 |
$S1E=15+39.256/60 |
$S2N=$S1N+( ... )/60000 |
$S2E=$S1E+( ... )/60000 |
$FN=$S2N+( ... )/60000 |
$FE=$S2E+( ... )/60000
{c:geo-end}
```

Listing 12: Codeblock-Vorlage zum Rechnen im „DDD.DDDDD°-Format“

```
{c:geo-start}
@[P1]Parkplatz (P) {CC|$S1N°|$S1E°}
@[S1]Station (S) {CC|$S2N°|$S2E°}
@[X1]Frage (X) {CC|$S3N°|$S3E°}
@[T1]Ausgangspunkt (T) {CC|$S4N°|$S4E°}
@[W1]Wegpunkt (W) {CC|$S5N°|$S5E°}
@[F1]Finale (F) {CC|$FN°|$FE°}
$S1N=48+40.340/60 |
$S1E=15+39.256/60 |
$S2N=$S1N+( )/60000 |
$S2E=$S1E+( )/60000 |
$S3N=$S2N+( )/60000 |
$S3E=$S2E+( )/60000 |
$S4N=$S3N+( )/60000 |
$S4E=$S3E+( )/60000 |
$S5N=$S4N+( )/60000 |
$S5E=$S4E+( )/60000 |
$FN=$S5N+( )/60000 |
$FE=$S5E+( )/60000
{c:geo-end}
```

Listing 13: Codeblock-Vorlage mit allen Wegpunkttypen im „DDD.DDDDD°-Format“

```
{c:geo-start}
@Station 1 (S) {CC|$S1N|$S1O}
@Station 2 (S) {CC|$S2N|$S2O}
@Station 3 (S) {CC|$S3N|$S3O}
@Station 4 (S) {CC|$S4N|$S4O}
@Station 5 (S) {CC|$S5N|$S5O}
@Finale (F) {CC|$S6N|$S6O}
$a= | $b= | $c= | $d=lv('') | $e=lv('') | $f= | $g= | $h= |
$i= | $k= | $m= | $n= | $o= | $p= | $x=lv('') | $y= |
$S1N=48+41.110/60 |
$S1O=14+45.335/60 |
$S2N=$S1N+(a*a)/60000 |
$S2O=$S1O+(a*4-5)/60000 |
$S3N=$S2N+(-b+c)/60000 |
$S3O=$S2O+(-b-c-7)/60000 |
$S4N=$S3N+(h-f-g+o)/60000 |
$S4O=$S3O+(-d)/60000 |
$S5N=$S4N+(c-e)/60000 |
$S5O=$S4O+(g-a+k+k-o)/60000 |
$S6N=$S5N+(p+c-o*f-m*e)/60000 |
$S6O=$S5O+(h-i+y-n-x+k-d-g+o)/60000
{c:geo-end}
```

Listing 14: „Xunde Runde“ GC1T9ZX

```
{c:geo-start}
@Final Location (F) {CC|$SFN|$SFO}
$A= | $B= | $C= | $D= | $E= | $F= | $G= | $H= | $I= | $J= | $K= | $L= | $M= | $N= | $P= |
$S01N=48+12.694/60 |
$S01O=14+51.524/60 |
$S01D=A*100 | $S01W=A*50 |
$S02N=$S01N+($S01D/1853*cos($S01W))/60 |
$S02O=$S01O+(($S01D*sin($S01W))/(1853*cos($S02N)))/60 |
$S02D=7B | $S02W=(B-A)*10+1 |
$S03N=$S02N+($S02D/1853*cos($S02W))/60 |
$S03O=$S02O+(($S02D*sin($S02W))/(1853*cos($S03N)))/60 |
$S03D=C0 | $S03W=C*10/2 |
$S04N=$S03N+($S03D/1853*cos($S03W))/60 |
$S04O=$S03O+(($S03D*sin($S03W))/(1853*cos($S04N)))/60 |
$S04D=A7D | $S04W=B*C |
$S05N=$S04N+($S04D/1853*cos($S04W))/60 |
$S05O=$S04O+(($S04D*sin($S04W))/(1853*cos($S05N)))/60 |
$S05D=242 | $S05W=274 |
$S06N=$S05N+($S05D/1853*cos($S05W))/60 |
$S06O=$S05O+(($S05D*sin($S05W))/(1853*cos($S06N)))/60 |
$S06D=ECF | $S06W=B*F+D |
$S07N=$S06N+($S06D/1853*cos($S06W))/60 |
$S07O=$S06O+(($S06D*sin($S06W))/(1853*cos($S07N)))/60 |
$S07D=EA0 | $S07W=DD(G-A) |
$S08N=$S07N+($S07D/1853*cos($S07W))/60 |
$S08O=$S07O+(($S07D*sin($S07W))/(1853*cos($S08N)))/60 |
$S08D=ECA | $S08W=DAF |
$S09N=$S08N+($S08D/1853*cos($S08W))/60 |
$S09O=$S08O+(($S08D*sin($S08W))/(1853*cos($S09N)))/60 |
$S09D=EDD | $S09W=C |
$S10N=$S09N+($S09D/1853*cos($S09W))/60 |
$S10O=$S09O+(($S09D*sin($S09W))/(1853*cos($S10N)))/60 |
$S10D=AGG | $S10W=EED |
$S11N=$S10N+($S10D/1853*cos($S10W))/60 |
$S11O=$S10O+(($S10D*sin($S10W))/(1853*cos($S11N)))/60 |
$S11D=KCE | $S11W=FI |
$S12N=$S11N+($S11D/1853*cos($S11W))/60 |
$S12O=$S11O+(($S11D*sin($S11W))/(1853*cos($S12N)))/60 |
$S12D=AID | $S12W=AAD |
$S13N=$S12N+($S12D/1853*cos($S12W))/60 |
$S13O=$S12O+(($S12D*sin($S12W))/(1853*cos($S13N)))/60 |
$S13D=HMK | $S13W=MOD |
$S14N=$S13N+($S13D/1853*cos($S13W))/60 |
$S14O=$S13O+(($S13D*sin($S13W))/(1853*cos($S14N)))/60 |
$S14D=IEE | $S14W=MDM |
$S15N=$S14N+($S14D/1853*cos($S14W))/60 |
$S15O=$S14O+(($S14D*sin($S14W))/(1853*cos($S15N)))/60 |
$S15D=(A+B+C+D+E+F+G+H+I+J+K+L+M+N+P)*N-(P+N+M+K) |
$S15W=(A+B+C+D+E+F+G+H+I+J+K+L+M+N+P)+(L-P-F-D/F) |
$SFN=$S15N+($S15D/1853*cos($S15W))/60 |
$SFO=$S15O+(($S15D*sin($S15W))/(1853*cos($SFN)))/60
{c:geo-end}
```

Listing 15: „Waypoint Challenge #1“ GC3N56E

```
{c:geo-start}
@Finale (F) {CC|N48°39.ABC'|E015°39.DEF'}
$A=cs() |
$B=roman('')-roman('') |
$C=cs('')|
$D= |
$E= |
$F=cs()
{c:geo-end}
```

Listing 16: „HORN iST VORN - Stadtpark“ GC5CWXF

```
{c:geo-start}
@Finale (F) {CC|N48°($ZSH)($ZSE-$ZSJ-$ZSA).($ZSE)($ZSI-$ZSD)($ZSI-$ZSJ)|
E15°($ZSG-$ZSH)($ZSF).($ZSC-$ZSD)($ZSB-$ZSH)($ZSC)}
$V01A=|$V01B=|$V01C=|$V01D=|$V01E=|$V01F=|$V01G=|
$V01X=if($V01A+$V01B+$V01C+$V01D+$V01E+$V01F+$V01G==35;'OK';'-')|
$V02A=|$V02B=|$V02C=|$V02D=|$V02E=|$V02F=|$V02G=|$V02H=|$V02I=|
$V02X=if($V02A+$V02B+$V02C+$V02D+$V02E+$V02F+$V02G+$V02H+$V02I==51;'OK';'-')|
$V03A=|$V03B=|$V03C=|$V03D=|$V03E=|
$V03X=if($V03A+$V03B+$V03C+$V03D+$V03E==12;'OK';'-')|
$V04A=|$V04B=|$V04C=|$V04D=|$V04E=|$V04F=|$V04G=|
$V04X=if($V04A+$V04B+$V04C+$V04D+$V04E+$V04F+$V04G==36;'OK';'-')|
$V05A=|$V05B=|$V05C=|$V05D=|$V05E=|$V05F=|$V05G=|$V05H=|$V05I=|$V05J=|
$V05X=if($V05A+$V05B+$V05C+$V05D+$V05E+$V05F+$V05G+$V05H+$V05I+$V05J==43;'OK';'-')|
$V06A=|$V06B=|$V06C=|$V06D=|
$V06X=if($V06A+$V06B+$V06C+$V06D==22;'OK';'-')|
$V07A=|$V07B=|$V07C=|$V07D=|$V07E=|$V07F=|
$V07X=if($V07A+$V07B+$V07C+$V07D+$V07E+$V07F==34;'OK';'-')|
$V08A=|$V08B=|$V08C=|$V08D=|$V08E=|
$V08X=if($V08A+$V08B+$V08C+$V08D+$V08E==20;'OK';'-')|
$V09A=|$V09B=|$V09C=|$V09D=|$V09E=|$V09F=|$V09G=|$V09H=|$V09I=|
$V09X=if($V09A+$V09B+$V09C+$V09D+$V09E+$V09F+$V09G+$V09H+$V09I==32;'OK';'-')|
$V10A=|$V10B=|$V10C=|$V10D=|$V10E=|$V10F=|$V10G=|$V10H=|
$V10X=if($V10A+$V10B+$V10C+$V10D+$V10E+$V10F+$V10G+$V10H==27;'OK';'-')|
$V11A=|$V11B=|$V11C=|$V11D=|$V11E=|
$V11X=if($V11A+$V11B+$V11C+$V11D+$V11E==18;'OK';'-')|
$V12A=|$V12B=|$V12C=|$V12D=|$V12E=|$V12F=|
$V12X=if($V12A+$V12B+$V12C+$V12D+$V12E+$V12F==31;'OK';'-')|
$V13A=|$V13B=|$V13C=|$V13D=|$V13E=|$V13F=|$V13G=|$V13H=|
$V13X=if($V13A+$V13B+$V13C+$V13D+$V13E+$V13F+$V13G+$V13H==57;'OK';'-')|
$V14A=|$V14B=|$V14C=|
$V14X=if($V14A+$V14B+$V14C==20;'OK';'-')|
$ZSA=ics($V01A+$V02D+$V03E+$V05C+$V06C+$V08C+$V09H+$V11A+$V12F+$V14B)|
$ZSB=ics($V01B+$V02E+$V04A+$V05D+$V06D+$V08D+$V09I+$V11B+$V13A+$V14C)|
$ZSC=ics($V01C+$V02F+$V04B+$V05E+$V07A+$V08E+$V10A+$V11C+$V13B)|
$ZSD=ics($V01D+$V02G+$V04C+$V05F+$V07B+$V09A+$V10B+$V11D+$V13C)|
$ZSE=ics($V01E+$V02H+$V04D+$V05G+$V07C+$V09B+$V10C+$V11E+$V13D)|
$ZSF=ics($V01F+$V02I+$V04E+$V05H+$V07D+$V09C+$V10D+$V12A+$V13E)|
$ZSG=ics($V01G+$V03A+$V04F+$V05I+$V07E+$V09D+$V10E+$V12B+$V13F)|
$ZSH=ics($V02A+$V03B+$V04G+$V05J+$V07F+$V09E+$V10F+$V12C+$V13G)|
$ZSI=ics($V02B+$V03C+$V05A+$V06A+$V08A+$V09F+$V10G+$V12D+$V13H)|
$ZSJ=ics($V02C+$V03D+$V05B+$V06B+$V08B+$V09G+$V10H+$V12E+$V14A)
{c:geo-end}
```

Listing 17: „Welterbesteig Wachau“ GC8PWHQ

```
{c:geo-start}
@[02]Die Abzweigung 42 (X) {CC|$S2N°|$S2E°}
@[03]Der Wegweiser (X) {CC|$S3N°|$S3E°}
@[04]Das Gehöft (X) {CC|$S4N°|$S4E°}
@[05]Die Tafel (X) {CC|$S5N°|$S5E°}
@[06]Der Aussichtspunkt (S) {CC|$S6N°|$S6E°}
@[FN]Final Location (F) {CC|$FN°|$FE°}
$A= | $B=len('') | $C=len('') | $D= | $E=len('') | $F=cs('') |
$S1N=48+30.657/60 |
$S1E=15+29.599/60 |
$S2N=$S1N+(-22*$A-7)/60000 |
$S2E=$S1E+(-6*$A+2)/60000 |
$S3N=$S2N+(-3*$A-5*$B-5)/60000 |
$S3E=$S2E+(+2*$A+3*$B+4)/60000 |
$S4N=$S3N+(-6*$A-10*$B-8*$C+3)/60000 |
$S4E=$S3E+(-1*$A+2*$B-2*$C+4)/60000 |
$S5N=$S4N+(-1*$A-1*$B-1*$C-1*$D-1)/60000 |
$S5E=$S4E+(-1*$A+2*$B-1*$C+1*$D+3)/60000 |
$S6N=$S5N+(+2*$A-1*$B+1*$C-1*$D-1*$E+3)/60000 |
$S6E=$S5E+(+1*$A-1*$B+1*$C-1*$D+3*$E-3)/60000 |
$FN=$S6N+(+5*$A+6*$B+8*$C+1*$D+8*$E+1*$F-2)/60000 |
$FE=$S6E+(+1*$A+1*$B+2*$C+2*$D+2*$E+2*$F)/60000
{c:geo-end}
```

Listing 18: „Donaublick“ GC8V75W

```
{c:geo-start}
@[H1]INITIUM (S) {v} N47 51.882 E15 51.474
@[H2]ARS NOVA (S) {v} {CC|N$S2N°|E$S2E°}
@[H3]CONGIARIUM (X) {v} {CC|N$S3N°|E$S3E°}
@[H4]GLADIUS (S) {v} {CC|N$S4N|E$S4E°}
@[H6]INSIGNES (S) {v} {CC|N$S6N°|E$S6E°}
@[H7]ANTE FINEM (S) {CC|N$S7N°|E$S7E°}
@VADEMECUM (S) {v} {CC|N$S5N°|E$S5E°}
@FINIS (F) {v} {CC|N$FN°|E$FE°}
$A=lv('') | $B=cs() | $C= | $D= | $E= | $F= | $G=cs() |
$S1N=47+51.881/60 |
$S1E=15+51.474/60 |
$S2N=$S1N+(2*$A)/60000 |
$S2E=$S1E+(14*$A-12)/60000 |
$S3N=$S2N+(4*$B-4)/60000 |
$S3E=$S2E+(10*$A+6*$B)/60000 |
$S4E=$S3E+(10*$A+10*$B+13*$C)/60000 |
$S4N=$S3N+(10*$A+20*$C)/60000 |
$S5N=$S4N-((40*$D-$A+$D)/60000) |
$S5E=$S4E-((50*$B+40*$D+4*$C)/60000) |
$S6N=$S5N+($E-2*$B+1)/60000 |
$S6E=$S5E+($E+11*$B)/60000 |
$S7N=$S6N+($F+11*$D+$C)/60000 |
$S7E=$S6E+(2*$F+6*$B)/60000 |
$FN=47+50.000/60+($E*$B-$A-$D*$C*3)/60000 |
$FE=15+50.000/60+($A*$F-$D*$F+$G-3)/60000
{c:geo-end}
```

Listing 19: „Heilige Aussichten“ GC12GW0

```
{c:geo-start}
@Finale (F) {CC|N48°2($ZSE).($ZSJ)($ZSG+$ZSH)($ZSB-$ZSC)|
E15°3($ZSE).($ZSA-$ZSI)($ZSG-$ZSJ)($ZSD-$ZSH)}
$V01A=|$V01B=|$V01C=|$V01D=|$V01E=|$V01F=|
$V01X=if($V01A+$V01B+$V01C+$V01D+$V01E+$V01F==27;'OK';'-')|
$V02A=|$V02B=|$V02C=|$V02D=|$V02E=|$V02F=|
$V02X=if($V02A+$V02B+$V02C+$V02D+$V02E+$V02F==22;'OK';'-')|
$V03A=|$V03B=|$V03C=|$V03D=|$V03E=|
$V03X=if($V03A+$V03B+$V03C+$V03D+$V03E==27;'OK';'-')|
$V04A=|$V04B=|$V04C=|$V04D=|$V04E=|$V04F=|$V04G=|$V04H=|
$V04X=if($V04A+$V04B+$V04C+$V04D+$V04E+$V04F+$V04G+$V04H==34;'OK';'-')|
$V05A=|$V05B=|$V05C=|$V05D=|$V05E=|$V05F=|$V05G=|$V05H=|$V05I=|
$V05X=if($V05A+$V05B+$V05C+$V05D+$V05E+$V05F+$V05G+$V05H+$V05I==35;'OK';'-')|
$V08A=|$V08B=|$V08C=|$V08D=|$V08E=|$V08F=|$V08G=|
$V08X=if($V08A+$V08B+$V08C+$V08D+$V08E+$V08F+$V08G==26;'OK';'-')|
$V09A=|$V09B=|$V09C=|$V09D=|$V09E=|$V09F=|
$V09X=if($V09A+$V09B+$V09C+$V09D+$V09E+$V09F==28;'OK';'-')|
$V10A=|$V10B=|$V10C=|$V10D=|$V10E=|
$V10X=if($V10A+$V10B+$V10C+$V10D+$V10E==27;'OK';'-')|
$V11A=|$V11B=|$V11C=|$V11D=|$V11E=|
$V11X=if($V11A+$V11B+$V11C+$V11D+$V11E==22;'OK';'-')|
$V13A=|$V13B=|$V13C=|$V13D=|$V13E=|$V13F=|
$V13G=|$V13H=|$V13I=|$V13J=|$V13K=|
$V13X=if($V13A+$V13B+$V13C+$V13D+$V13E+$V13F+$V13G+$V13H+$V13I+$V13J+$V13K==55;'OK';'-')|
$V14A=|$V14B=|$V14C=|$V14D=|$V14E=|$V14F=|
$V14G=|$V14H=|$V14I=|$V14J=|$V14K=|$V14L=|
$V14X=if($V14A+$V14B+$V14C+$V14D+$V14E+$V14F+$V14G+$V14H+$V14I+$V14J+$V14K+$V14L==50;'OK';'-')|
$V15A=|$V15B=|$V15C=|$V15D=|$V15E=|$V15F=|$V15G=|$V15H=|
$V15X=if($V15A+$V15B+$V15C+$V15D+$V15E+$V15F+$V15G+$V15H==43;'OK';'-')|
$ZSA=ics($V01A+$V02E+$V04D+$V05F+$V08G+$V10D+$V13D+$V14C+$V15A)|
$ZSB=ics($V01B+$V02F+$V04E+$V05G+$V09A+$V10E+$V13E+$V14D+$V15B)|
$ZSC=ics($V01C+$V03A+$V04F+$V05H+$V09B+$V11A+$V13F+$V14E+$V15C)|
$ZSD=ics($V01D+$V03B+$V04G+$V05I+$V09C+$V11B+$V13G+$V14F+$V15D)|
$ZSE=ics($V01E+$V03C+$V04H+$V08A+$V09D+$V11C+$V13H+$V14G+$V15E)|
$ZSF=ics($V01F+$V03D+$V05A+$V08B+$V09E+$V11D+$V13I+$V14H+$V15F)|
$ZSG=ics($V02A+$V03E+$V05B+$V08C+$V09F+$V11E+$V13J+$V14I+$V15G)|
$ZSH=ics($V02B+$V04A+$V05C+$V08D+$V10A+$V13A+$V13K+$V14J+$V15H)|
$ZSI=ics($V02C+$V04B+$V05D+$V08E+$V10B+$V13B+$V14A+$V14K)|
$ZSJ=ics($V02D+$V04C+$V05E+$V08F+$V10C+$V13C+$V14B+$V14L)
{c:geo-end}
```

Listing 20: „Kremstal Donau Weitwanderweg“ GC960XM

```
{c:geo-start}
@[FN]Final Location (F) {CC|N48°$CacheN|E15°$CacheE}
@[R1]Reference 1 (W) {CC|N48°$Ref1N|E15°$Ref1E}
@[R2]Reference 2 (W) {CC|N48°$Ref2N|E15°$Ref2E}
@[S1]Stage 1 (X) {CC|N48°$Stage1N|E15°$Stage1E}
$Stage1E= |
$Stage1N= |
$A=len('') |
$B=bww('') |
$C= |
$S=$A+$B+$C |
$CacheE=$Stage1E-$S/1000 |
$CacheN=$Stage1N-($S-4*$C+1)/1000 |
$Ref1E=$Stage1E-($S+$C-$A+2)/1000 |
$Ref1N=$Stage1N-($S-3*$C-2)/1000 |
$Ref2E=$Stage1E-($S-4*$C-3)/1000 |
$Ref2N=$Stage1N-($S-5*$A-2*$C+1)/1000
{c:geo-end}
```

Listing 21: „Roman Numerals“ GCA2QQE

```
{c:geo-start}
@Final (F){CC|N48°20.(A*G)+(B*E)-(D*I)-C-8|E16°3.(C*D)+(F*H)-(C*G)+F+I+2}
$A1=|$B1=|$C1=|$D1=|$E1=|$F1=|$G1=|$H1=|$I1=|$J1=|$K1=|$L1=|$M1=|$N1=|$O1=|$P1=|$Q1=|$R1=|$S1=|
$T1=|$U1=|$V1=|$W1=|$X1=|$Y1=|$Z1=|$AA1=|$AB1=|$AC1=|$AD1=|$AE1=|$AF1=|
$A2=|$B2=|$C2=|$D2=|$E2=|$F2=|$G2=|$H2=|$I2=|$J2=|$K2=|$L2=|$M2=|$N2=|$O2=|$P2=|$Q2=|$R2=|$S2=|
$T2=|$U2=|$V2=|$W2=|$X2=|$Y2=|$Z2=|$AA2=|$AB2=|$AC2=|$AD2=|$AE2=|$AF2=|$AG2=|
$A3=|$B3=|$C3=|$D3=|$E3=|$F3=|$G3=|$H3=|$I3=|$J3=|$K3=|$L3=|$M3=|$N3=|$O3=|$P3=|$Q3=|$R3=|$S3=|
$T3=|$U3=|$V3=|$W3=|$X3=|$Y3=|$Z3=|$AA3=|$AB3=|
$A4=|$B4=|$C4=|$D4=|$E4=|$F4=|$G4=|$H4=|$I4=|$J4=|$K4=|$L4=|$M4=|$N4=|$O4=|$P4=|$Q4=|$R4=|$S4=|
$T4=|$U4=|$V4=|$W4=|$X4=|$Y4=|$Z4=|$AA4=|$AB4=|$AC4=|$AD4=|$AE4=|$AF4=|
$A5=|$B5=|$C5=|$D5=|$E5=|$F5=|$G5=|$H5=|$I5=|$J5=|$K5=|$L5=|$M5=|$N5=|$O5=|$P5=|$Q5=|$R5=|$S5=|
$T5=|$U5=|$V5=|
$A6=|$B6=|$C6=|$D6=|$E6=|$F6=|$G6=|$H6=|$I6=|$J6=|$K6=|$L6=|$M6=|$N6=|$O6=|$P6=|$Q6=|$R6=|$S6=|
$T6=|$U6=|$V6=|$W6=|
$A7=|$B7=|$C7=|$D7=|$E7=|$F7=|$G7=|$H7=|$I7=|$J7=|$K7=|$L7=|$M7=|$N7=|$O7=|$P7=|$Q7=|$R7=|$S7=|
$T7=|$U7=|
$A8=|$B8=|$C8=|$D8=|$E8=|$F8=|$G8=|$H8=|$I8=|$J8=|$K8=|$L8=|$M8=|$N8=|$O8=|$P8=|$Q8=|$R8=|$S8=|
$T8=|$U8=|$V8=|
$A9=|$B9=|$C9=|$D9=|$E9=|$F9=|$G9=|$H9=|$I9=|$J9=|$K9=|$L9=|$M9=|$N9=|$O9=|$P9=|$Q9=|$R9=|$S9=|
$T9=|$U9=|
$A=cs($A1+$B1+$C1+$D1+$E1+$F1+$G1+$H1+$I1+$J1+$K1+$L1+$M1+$N1+$O1+$P1+$Q1+$R1+$S1+$T1+$U1+$V1+$W1+
     $X1+$Y1+$Z1+$AA1+$AB1+$AC1+$AD1+$AE1+$AF1)|
$B=cs($A2+$B2+$C2+$D2+$E2+$F2+$G2+$H2+$I2+$J2+$K2+$L2+$M2+$N2+$O2+$P2+$Q2+$R2+$S2+$T2+$U2+$V2+$W2+
     $X2+$Y2+$Z2+$AA2+$AB2+$AC2+$AD2+$AE2+$AF2+$AG2)|
$C=cs($A3+$B3+$C3+$D3+$E3+$F3+$G3+$H3+$I3+$J3+$K3+$L3+$M3+$N3+$O3+$P3+$Q3+$R3+$S3+$T3+$U3+$V3+$W3+
     $X3+$Y3+$Z3+$AA3+$AB3)|
$D=cs($A4+$B4+$C4+$D4+$E4+$F4+$G4+$H4+$I4+$J4+$K4+$L4+$M4+$N4+$O4+$P4+$Q4+$R4+$S4+$T4+$U4+$V4+$W4+
     $X4+$Y4+$Z4+$AA4+$AB4+$AC4+$AD4+$AE4+$AF4)|
$E=cs($A5+$B5+$C5+$D5+$E5+$F5+$G5+$H5+$I5+$J5+$K5+$L5+$M5+$N5+$O5+$P5+$Q5+$R5+$S5+$T5+$U5+$V5)|
$F=cs($A6+$B6+$C6+$D6+$E6+$F6+$G6+$H6+$I6+$J6+$K6+$L6+$M6+$N6+$O6+$P6+$Q6+$R6+$S6+$T6+$U6+$V6+$W6)
     |
$G=cs($A7+$B7+$C7+$D7+$E7+$F7+$G7+$H7+$I7+$J7+$K7+$L7+$M7+$N7+$O7+$P7+$Q7+$R7+$S7+$T7+$U7)|
$H=cs($A8+$B8+$C8+$D8+$E8+$F8+$G8+$H8+$I8+$J8+$K8+$L8+$M8+$N8+$O8+$P8+$Q8+$R8+$S8+$T8+$U8+$V8)|
$I=cs($A9+$B9+$C9+$D9+$E9+$F9+$G9+$H9+$I9+$J9+$K9+$L9+$M9+$N9+$O9+$P9+$Q9+$R9+$S9+$T9+$U9)
{c:geo-end}
```

Listing 22: „Großer Tullnerfelder Rundwanderweg 475/675“ GCABJDB