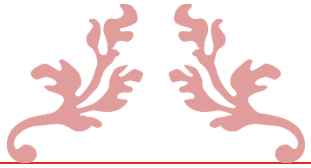


IBAN RECHNER

Dokumentation

28. JULI 2019

xxxxxxxxxxxxxx

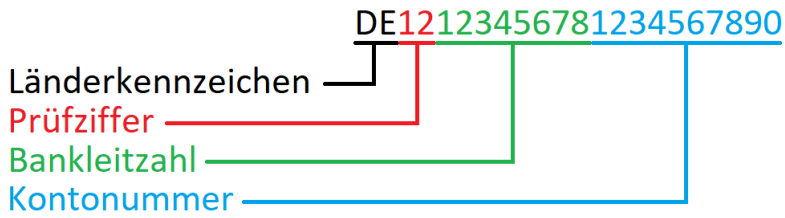
INHALTSVERZEICHNIS

NUTZEN

Der IBAN Rechner hat zwei verschiedene Funktion: Das Berechnen der IBAN anhand einer Kontonummer und einer Bankleitzahl und das Überprüfen einer IBAN auf Richtigkeit unter Zuhilfenahme der Prüfziffer. Dieser IBAN Rechner ist nur nutzbar mit deutschen IBANs. Das IBAN Format wird zwar in 27 Ländern benutzt, wird aber in jedem Land unterschiedlich zusammengesetzt, was die Programmierung dieses Programms immens verkompliziert hätte.

IBAN

AUFBAU



Die ersten zwei Buchstaben enthalten das Länderkennzeichen. Die zwei darauffolgenden Ziffern sind die Prüfziffern. Sie befinden sich im Normalfall zwischen „02“ und „98“. (Da in diesem Programm die Richtigkeit der Bankleitzahl und Kontonummer nicht kontrolliert wird, kann aber es aber auch zu „00“, „01“ und „99“ kommen.) Die nächsten 8 Ziffern sind die Bankleitzahl. Diese ist in Deutschland generell achtstellig. Die letzten 10 Ziffern sind die Kontonummer. Bei älteren oder speziellen Konten kann es sein, dass diese sich aus weniger Ziffern zusammensetzen. In diesem Fall werden vor die Kontonummer führende Nullen eingesetzt, damit man auf 10 Ziffern kommt. Dies ist wichtig für die Berechnung der Prüfziffer.

BERECHNUNG DER PRÜFZIFFER

Kontonummern, die nicht zehnstellig sind, werden mit führenden Nullen erweitert. Der Ländercode DE wird in eine Zahl umgewandelt, indem man die einzelnen Stellen im Alphabet nimmt und mit 9 addiert: D E -> 4 5 -> 13 14.

Beispiel: Das Spendenkonto von UNICEF Deutschland
Kontonummer 30 0000
Bankleitzahl: 370 205 00

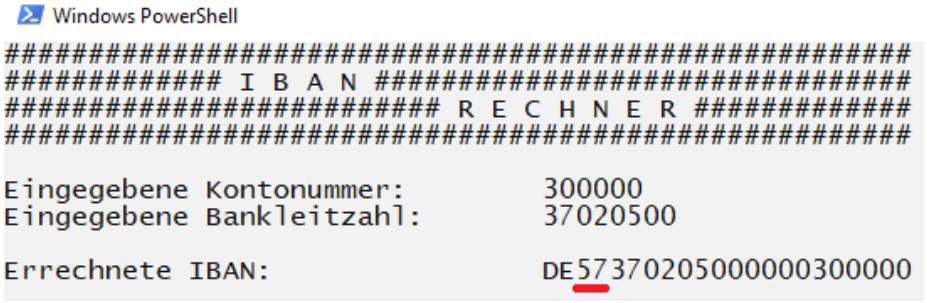
Aus Bankleitzahl, Kontonummer, Länderkennzeichen und einer auf 00 gesetzten Prüfziffer wird eine Zahl gebildet:

370205000000300000131400

Diese Zahl wird jetzt ganzzahlig durch 97 geteilt (Modulo 97). Der Rest wird anschließend von 98 abgezogen. Sollte das Ergebnis einstellig sein, wird es um eine führende Null erweitert.

370205000000300000131400 % 97 = 41
98 – 41 = 57

Die Prüfziffer ist nach dieser Rechnung 57. Zur Probe:



QUELLCODE, METHOD IN MADNESS

Einzelne Funktionen und Abläufe werden in der Reihenfolge ihrer Benutzung erklärt. Außerdem wurde dieses Programm auf Windows geschrieben, durch die Verwendung von conio.h kann es nicht auf Unix basierten Systemen kompiliert werden.

INKLUDIERTE FILES

```
#include "functions.c"
#include "calculator.c"
#include "validator.c"
```

Das ganze main.c File wurde etwas unübersichtlich, deswegen wurden einzelne Funktionen und die beiden Hauptfunktionen des Programmes auf einzelne .c Dateien aufgeteilt.

HEADER() FUNCTIONS.C ZEILE 10

Um die User Experience etwas zu verschönern, wird die Funktion header() ausgeführt. Sie sendet zuerst cls oder clear an das Terminal, um alle vorherigen Inhalte zu löschen und gibt dann auf vier Zeilen einen Header aus, der den Programmnamen enthält. (Siehe Screenshot oben)

INT MODUS = MODUSWAHL() MAIN.C ZEILE 22

In Präparation auf einen Switch namens Modus wird dieser Integer mit Hilfe der Funktion moduswahl() initiiert.

MODUSWAHL() FUNCTIONS.C ZEILE 20

Der User wird gebeten eine Auswahl zu treffen: 1 für Berechnung der IBAN, 2 für Prüfung der IBAN, ESC zum Abbrechen. Mit Hilfe von getch() wird der Tastenanschlag auf char wahl gespeichert. Wenn wahl 27 entspricht, also Escape, wird die Funktion mit -1 returned.

Ein neuer Integer namens newWahl wird nun durch atoi(wahl) initiiert. Mit diesem neuem Integer wird ein Switch bestückt. Wenn die Auswahl 1 war, wird 1 returned. Wenn die Auswahl 2 war, wird 2 returned. Wenn einer der beiden Cases nicht erfüllt wird, greift default, gibt eine Fehlermeldung aus („Wahlmöglichkeiten 1, 2, oder ESC zum Abbrechen.“) und ruft sich selbst wieder auf -> Rekursive Programmierung.

SWITCH (MODUS) MAIN.C ZEILE 24

In diesem Switch werden die beiden möglichen Funktionen IBANcalc() oder IBANVval() basierend auf die Werte die moduswahl() returned hat, aufgerufen. Wenn durch moduswahl() -1 durch Betätigung der Escape Taste zurückkommt, greift der default Case, druckt eine Fehlermeldung („Operation abgebrochen.“) und das Programm wird beendet.

CASE 1: IBANCALC() CALCULATOR.C

Als erstes wird durch eine While-Schleife die Bankleitzahl eingegeben. Die Bedingung der While-Schleife ist blzChecked == 0. Der User wird gebeten, eine achtstellige Bankleitzahl einzugeben. Diese wird durch gets(blz) gespeichert. Anschließend wird durch blzChecked = checkBLZ(blz) die Funktion checkBLZ() aufgerufen. Wenn checkBLZ eine 1 returned, ist die While-Schleife gebrochen und das Programm kann weiter gehen, ansonsten wird es wieder und wieder ausgeführt, bis eine achtstellige Bankleitzahl eingegeben wird.

CHECKBLZ() FUNCTIONS.C ZEILE 51

Diese Funktion überprüft die Bankleitzahl auf zwei Eigenschaften. Zuerst, ob die Bankleitzahl Buchstaben enthält und dann, ob die Bankleitzahl 8 Ziffern lang ist oder nicht.

Die erste Eigenschaft wird in einer While-Schleife überprüft. Solange blz[i] (int i = 0) existiert, wird eine If-Anweisung ausgeführt, die abfragt, ob der aktuelle Character alphabetisch ist oder nicht. Das Ganze wird realisiert mit der Funktion isalpha(). Wenn diese Bedingung gegeben ist, wird eine Fehlermeldung gedruckt („Die eingegebene Bankleitzahl enthält Buchstaben, dies ist nicht zulässig.“) und der Wert 0 returned. Die Schleife blzChecked sorgt in diesem Fall dafür, dass die Bankleitzahl erneut eingegeben werden muss. Nach jedem überstandenen Durchgang wird i um 1 erhöht, damit der nächste Character überprüft werden kann.

Als nächstes wird die Länge des Strings überprüft mit if (strlen(blz) == 8). Wenn auch dieser Test abgeschlossen ist, wird der Wert 1 returned und die Bedingung blzChecked ist erfüllt.

Falls dieser zweite Test nicht erfolgreich ist, wird mit else eine Fehlermeldung gedruckt („Die eingegebene Bankleitzahl ist nicht 8 Ziffern lang, dies ist nicht zulässig.“) und der Wert 0 returned. Der User kann die Bankleitzahl erneut eingeben.

ZURÜCK ZU IBANCALC() — CALCULATOR.C ZEILE 25

Als nächstes wird die Kontonummer im gleichen Schema wie die Bankleitzahl eingelesen. While-Schleife solange ktNrChecked == 0 ist, gets(ktNr) um den Kontonummer String einzulesen und es geht weiter mit ktNrChecked = checkKonto(ktNr).

CHECKKONTO() — FUNCTIONS.C ZEILE 84

Als erstes wird hier, genauso wie bei checkBLZ(), überprüft, ob sich ein Buchstabe eingeschlichen hat.

Da Kontonummern weniger als 10 Ziffern haben können, muss hier umgedacht werden. Wenn die Kontonummer weniger oder gleich 10 Stellen hat, wird als erstes ein neuer char base[] = "0000000000" initiiert. Dieser String wird zum Erweitern der Kontonummer mit führenden Nullen benutzt. Zunächst wird überprüft, wie groß der vorhandene String ktNr ist, dieser Wert wird dann von 10 abgezogen und als int leadingZero deklariert, um die Anzahl der benötigten führenden Nullen festzulegen.

Nun wird mit der Funktion strcpy() die Kontonummer auf den String base kopiert und zwar ab der Stelle, die davor in leadingZero gespeichert worden ist. Abschließend wird der jetzt neu bestückte String base wiederum mit strcpy() auf ktNr kopiert und 1 returned, damit die Funktion beendet wird und die Bedingung ktNrChecked == 0 gebrochen wird.

Falls die Kontonummer mehr als 10 Stellen hatte, kommt man zu einem else-Fall, der eine Fehlermeldung druckt und wandert mit einem return 0 wieder zurück, um die Kontonummer neu einzugeben.

CHAR IBANROH — CALCULATOR.C ZEILE 34

Als nächstes wird IBANroh in drei Durchgängen mit blz, ktNr, und dem char "land" zusammengeführt. "land" beinhaltet nicht nur das Länderkennzeichen, sondern auch die auf 00 gesetzte Prüfziffer, alles in numerischer Form ("131400").

ZAHL ZU LANG

IBANroh ist nun mit 24 Stellen zu lang für einen Integer und kann somit nicht berechnet werden. Als Workaround muss IBANroh aufgeteilt und in mehreren Schritten berechnet werden.

AUFTEILEN VON IBANROH — CALCULATOR.C ZEILE 39

Dieser Teil wurde inspiriert von einem Tutorial auf techcrashcourse.com namens „C Program to Divide a String into Two Equal Strings“.

Das Tutorial geht davon aus, dass der User einen x-beliebigen String eingibt. Die Länge wird dann mit strlen() ausgelesen und durch zwei geteilt, damit man in zwei For-Schleifen mit zwei Variablen namens mid und length arbeiten kann.

```
length = strlen(inputString);
mid = length/2;
for(i = 0; i < mid; i++)
{
    leftHalf[i]= inputString[i];
}
leftHalf[i] = '\0';
for(i = mid, k = 0; i <= length; i++, k++)
{
    rightHalf[k]= inputString[i];
}
```

Da IBANroh immer 24 Stellen hat, kann auf diesen Schritt verzichtet werden. Die For-Schleifen bekommen feste Parameter um sicherzustellen, dass die einzelnen IBAN Blöcke korrekt getrennt werden. (4 Blöcke à 6 Ziffern -> 0 bis 5, 6 bis 11, 12 bis 17, 18 bis 23)

Beispiel von einem Durchlauf:
IBANroh="370205000000300000131400" (Unicef Beispiel von oben)

i	0	1	2	3	4	5	6
Durchgang	1	2	3	4	5	6	
temp1	3	7	0	2	0	5	

```
for(i = 0; i < 6; i++)
{
    temp1[i] = IBANroh[i];
}
temp1[i] = '\0';
```

Anschließend kommt noch \0 dazu. temp1 ist nun 370205\0.

MODULO BERECHNUNG — CALCULATOR.C ZEILE 64

Nun haben wir 4 Strings (temp1, temp2, temp3, temp4). Da C nicht mit Strings rechnen kann, müssen diese erst mit Hilfe der Funktion atoi() umgewandelt werden.

Der erste Block wird mit Modulo 97 berechnet, mit 10^6 multipliziert und mit dem zweiten Block addiert. Der zweite Block wird mit Modulo 97 berechnet, mit 10^6 multipliziert und mit dem dritten Block addiert. Der dritte Block wird mit Modulo 97 berechnet, mit 10^6 multipliziert und mit dem vierten Block addiert. Der vierte Block wird mit Modulo 97 berechnet und zum Schluss von 98 abgezogen. Das Resultat ist die Prüfziffer.

```
iban1 %= 97;
iban2 += (iban1*1000000);
iban2 %= 97;
iban3 += (iban2*1000000);
iban3 %= 97;
iban4 += (iban3*1000000);
iban4 %= 97;
iban4 = 98-iban4;
```

OUTPUT — CALCULATOR.C ZEILE 84

Abschließend werden die eingegebene Kontonummer, Bankleitzahl und die zusammengesetzte IBAN mit errechneter Prüfziffer ausgegeben.

CASE 2: IBANVAL() — VALIDATOR.C

Als erstes wird der User gebeten, eine IBAN einzugeben. Mit Hilfe einer While-Schleife wird zunächst die Länge des eingegebenen Strings überprüft. Wenn der String zu kurz ist, kommt eine Fehlermeldung („Fehler: Die eingegebene IBAN ist zu kurz, dies ist nicht zulässig.“). Wenn der String zu lang ist, kommt ebenfalls eine Fehlermeldung („Fehler: Die eingegebene IBAN ist zu lang, dies ist nicht zulässig.“). Wenn die Länge korrekt ist, bricht die While-Schleife und es geht weiter.

AUFTEILEN DER IBAN — VALIDATOR.C ZEILE 47

Mit der gleichen Methode, die auch davor schon benutzt worden ist, wird die eingegebene IBAN auf 4 Stücke aufgeteilt: Länderkennzeichen, Prüfziffer, Bankleitzahl und Kontonummer.

CHECKIBAN(CC, PZ, BLZ, KTNR) — FUNCTIONS.C ZEILE 124

Die vier Strings werden nun in checkIBAN nach dem gleichen Prinzip wie checkKtNr und checkBLZ. Die Eingabe der IBAN und Überprüfung der Länge befinden sich in einer While-Schleife, die als Bedingung ibanChecked == 0 hat. Wenn in checkIBAN alle Prüfungen bestanden worden sind, wird eine 1 returned und die Schleife ist beendet. Bei Nichtbestehen der einzelnen Prüfungen wird immer eine 0 returned und der User muss die IBAN erneut eingeben.

ÜBERPRÜFUNG VON CC — FUNCTIONS.C ZEILE 133

Der String cc wird mithilfe von strcmp() mit dem erlaubten Länderkennzeichen „DE“ verglichen. Sofern der User dieses klein geschrieben hat, wird die Funktion strup() pauschal benutzt, um cc beim Vergleich als Großbuchstaben zu verwenden. Alternativ hätte man auch || verwenden können, um einmal mit „de“ und einmal mit „DE“ zu vergleichen.

ÜBERPRÜFUNG AUF BUCHSTABEN — FUNCTIONS.C ZEILE 142

Falls der User hinter dem Länderkennzeichen noch weitere Buchstaben geschrieben hat, wird wieder mit Hilfe der Funktion isalpha() gearbeitet. Prüfziffer, Bankleitzahl und Kontonummer werden mit strcat() zusammengeführt und anschließend, wie bereits in checkBLZ() erklärt, auf Buchstaben kontrolliert. Wenn Buchstaben gefunden werden, wird eine Fehlermeldung (Fehler: Die eingegebene IBAN enthält nach dem Ländercode Buchstaben) gedruckt und 0 returned. Wenn die Überprüfung nichts gefunden hat, wird 1 returned und die While-Schleife bricht.

BERECHNUNG DER KORREKTEN PRÜFZIFFER — VALIDATOR.C ZEILE 75

Nun geht es weiter wie auch in der Funktion ibanCALC(). Bankleitzahl, Kontonummer und Länderkennzeichen mit auf 00 gestellter Prüfziffer werden miteinander verbunden, in vier gleich große Teile aufgeteilt und mit Modulo 97 berechnet.

Abschließend wird überprüft, ob die neu errechnete Prüfziffer die gleiche ist, die der User eingegeben hat und mit if/else das entsprechende Resultat der Überprüfung mitgeteilt.

QUELLENANGABEN

1. https://www.sparkasse.de/geld-leichter-verstehen/i/ist-die-iban-aufgebaut.html
2. https://de.wikipedia.org/wiki/Internationale_Bankkontonummer
3. https://www.iban-rechner.de/calculation.html
4. https://www.techcrashcourse.com/2016/05/c-program-to-split-string-into-two-equal-strings.html