Teil 4: Fehler im Code finden / Debuggen

Nutzen Sie bitte ausschließlich die Programmierumgebung VSC

4.1 Datentyp float und der Debugger

a) Schreiben Sie das folgende C-Programm und starten es im Debugmodus.

```
float x = 0.0;
x = 1 - 10 * (1 - 0.9);
if (x < 0)
{
     printf("x < %0.2lf", x);
}
else if (x > 0)
{
     printf("x > %0.2lf", x);
}
else
{
     printf("x == %0.2lf", x);
}
```

Was wird nach der Ausführung des Programms ausgegeben? Wie kommt es zu dieser Ausgabe? Erstellen Sie im Debugmodus einen Screenshot der Variablen x bezüglich seines Wertes nach der Zuweisung.

b) Schreiben Sie ein C-Programm, das eine short Zahl `sNumber` deklariert und von der Tastatur eine `int`-Zahl `iNumber` einliest. Anschließend soll wie folgt eine Zuweisung mit einer impliziten Typumwandlung (TypeCast) erfolgen:

```
sNumber=iNumber;
```

Was wird nach der Ausführung des Programms ausgegeben? Wie kommt es zu dieser Ausgabe? Im Debugmodus, erstellen Sie einen Screenshot der Werte der Variablen `iNumber` und der Variablen `sNumber` nach dem impliziten TypeCast.

1. `iNumber` = 11500

2. `iNumber` = 1114112

3. `iNumber` = 17892934

Was wird nach der Ausführung des Programms ausgegeben? Wie kommt es zu dieser Ausgabe? Im Debugmodus, erstellen Sie einen Screenshot der Variablen `iNumber` und der Variablen `sNumber` bezüglich seines Wertes nach dem impliziten cast.

c) Schreiben Sie ein C-Programm, das von der Tastatur zwei short-Zahlen `Kapital` und `Zins` einliest und anschließend den Zinsfuß berechnet:

```
p = (Zins * 100); //Zeile 1
p = p / Kapital;  //Zeile 2
```

Das Programm soll den Benutzer auffordern, einen Wert für `Kapital` und `Zins` einzugeben. Bei der Ausgabe soll die Rechenoperation und das Ergebnis angezeigt werden. Würde man

beispielsweise für `Kapital` 10000 und für `Zins` 400 eingegeben, so soll die Ausgabe wie folgt aussehen:

```
P = (400 * 100) / 10000 = 4
```

Testen Sie Ihr Programm im Debugmodus auch für Kapital = 320 , Zins = 50 und Kapital = 4500 , Zins = 100.

Was wird nach der Ausführung des Programms ausgegeben? Wie kommt es zu dieser Ausgabe? Im Debugmodus erstellen Sie einen Screenshot der Variablen p, Zins und Kapital bezüglich seines Wertes in Zeile 1 und Zeile 2.

4.2 Debugging von Float Ungenauigkeiten

Gegeben ist folgender Programmcode:

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: void foo()
5: {
6:     int i;
7:     for (i = 0; i < 10; i++)
8:     {
9:        printf("%d\t\t",i);
10:    }
11: }
12:
13: int main()
14: {
15:     foo();
16:     return 0;
17: }
```

Achtung: Dieses Beispiel setzt in gewissem Maße voraus, dass Sie wissen, was eine **Funktion** ist. Da dies im Skript noch nicht ausführlich behandelt wurde, müssen Sie an dieser Stelle nur wissen, dass eine Funktion in C eine in sich geschlossene Codeeinheit darstellt, die eine spezifische Aufgabe ausführt und z.B. in der main aufgerufen werden kann, um diese Aufgabe auszuführen. Hier ist der Name dieser Funktion `foo()`. Dies ist nur ein Platzhalter, da die Funktion nicht anders sinnstiftend beschrieben werden kann/muss.

Das Programm soll die Genauigkeit der Darstellung von Gleitkommatypen demonstrieren. Führen Sie folgende Schritte aus:

- Ändern Sie den Datentyp der Zählvariable `i` von `int` auf `float`. `i` wird mit `0.0` initialisiert. Die Durchlauf-Bedingung wird von `i < 10` auf `i != 1.0` geändert. Die Zählvariable wird nach jedem Schleifendurchgang um `0.1` erhöht.
- Das Formatierungszeichen der `printf()`-Funktion muss für float angepasst werden.
- Warum tritt die Abbruchbedingung nie ein? Wie nennt man den Effekt bei der Programmierung, wenn die Abbruchbedingung nicht definiert ist oder nicht eintreten kann?
- Setzen Sie in der Funktion `foo()` sogenannte Haltepunkte (**breakpoints**) in den Zeilen 7 (for-Schleife) und 9 (Bildschirmausgabe). **Tipp**: Das Setzen der Breakpoints

geht durch Anklicken in der Höhe des zugehörigen Sourcecodezeile links neben der Zeilennummer. Der Breakpoint wird durch einen roten Kreis symbolisiert.

- Starten Sie den Debugger und führen Programm schrittweise bis `1.300000` aus (mit Prozedurschritt oder Tastaturkürzel ***[F10]***). Dabei zeigt das gelbe Dreieck rechts neben der Zeilennummer die aktuelle Sourcecodezeile, die als nächstes ablaufen soll.
- Beobachten Sie die Ausgaben im Konsole-Fenster und inspizieren Sie die Werte im Variablen Bereich. .
- Beenden Sie den Debugger und modifizieren Sie das Formatierungszeichen in der `printf()`-Funktion (Zeile 9) wie folgt: `%1.12f`.
- Starten Sie den Debugger erneut und führen Sie das Programm schrittweise aus.
- Beobachten Sie die Ausgaben im Konsole-Fenster und inspizieren Sie die Werte im Variablen Bereich. .
- Erläutern Sie kurz und präzise Ihre Beobachtungsergebnisse bzgl. der Ausgabe im Konsole-Fensters und der Werte der Variablen:
- Da die Zahl `0.1` binär periodisch ist `(0,0001100110011001100110011001100110011001100110011001100...)`, kann der Wert `1.0` nicht exakt dargestellt werden. Im Konsole-Fenster sieht man das erstmal nicht. Aber im Fenster Watches kann man erkennen, dass die Zählvariable i einige Stellen nach dem Komma eine Ungenauigkeit aufweist. Erst mit der Modifizierung der printf()-Funktion: `printf("%1.12f\t\t",i);`
  kann man diese Ungenauigkeit im Konsole-Fenster erkennen.

4.3 Programm verstehen und debuggen

Gegeben ist folgendes Programm:

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	int x, y, e, z, r, b, i;

	printf("1. Zahl eingeben:");
	scanf("%d", &x);

	printf("2. Zahl eingeben:");
	scanf("%d", &y);

	e = x;
	z = y;

	while (z != 0) {
		r = e % z;
		e = z;
		z = r;
	}

	printf("\nDas Ergebnis für %d und %d ist: %d\n", x,y,e);

	printf("\nBitte andere Basis angeben:\n");
	scanf("%d", &b);

	printf("\nErgebnis für %d und %d ist: ", x, y);

	for (i = 0; e > 0; i++)
```

```
	{
		char c;
		int r = e % b;
		e = e / b;
		if (r < 10)
			c = r + '0';
		else
			c = r + 'A' - 10;

		if (i > 0)
			printf (" + ");

		printf ("%c * %d^%d", c, b, i);
	}
	printf ("\n");

	return EXIT_SUCCESS;
}
```

Das Programm hat leider gravierende Mängel: es ist nicht kommentiert, die Variablennamen sind nicht aussagekräftig und die Ausgaben sind zu allgemein formuliert. Bitte kopieren Sie das Programm in ihre Entwicklungsumgebung und finden Sie heraus, was das Programm berechnet. Es werden zwei Ergebnisse ausgegeben für die Zahlen x und y. Was bedeutet die Ergebnisse?

Nachdem Sie den Sinn des Programms verstanden haben, kommentieren Sie das Programm, benennen Sie Variablennamen sinnvoll um und gestalten Sie die Ausgaben des Programmes so, dass der Benutzer auch weiß, was das Programm berechnet. Schauen Sie sich dazu auch die Programmierrichtlinien an.

Zur Lösung des Problems sollen Sie den Debugger verwenden. Setzen Sie mindestens zwei Breakpoints: Einen auf die erste Anweisung in der while Schleife, und einen auf die erste Anweisung in der for-Schleife. Inspizieren Sie die Inhalte der Variablen und beobachten Sie die Änderungen in jedem Schleifendurchgang.

Führen Sie im Debugger Einzelschritte und mehrere Schritte bis zum nächsten Breakpoint durch. Während der Übung sollten Sie den Debugger vorführen können.

4.4 Fehler im Code finden

Nutzt Inhalte von https://www.informatik.uni-leipzig.de

Alle folgenden Programme enthalten syntaktische oder logische Fehler. Bei einigen Programmen erhalten Sie Fehlermeldungen beim Erstellen der ausführbaren Datei, bei den logischen Fehlern erhalten Sie keine Fehlermeldung. Finden Sie die Fehler und machen einen Screenshot der Fehlermeldungen.

a)

```
int k, i = 0;
for( i = 0; i < 10; i++ )
{
    k = k + 1;
}
printf("%d", k);
```

b)

```
int a = 0;
int b = 0;
sum = a + b;
printf( "sum: %d", sum );
```

c)

```
#include <stdio.h>
int main () {
    int a = 0;
    scanf( "%d", a );
    printf( "Der Wert von a: %d", a );
    return 0;
}
```

d)

```
int k = 0;
for(int i = 0; i < 10; ++i ) {
    k =+ 1;  // k soll hoch zaehlen
}
```

e)

```
#include <stdio.h>
#define VALUE 5;

int main(void) {
    int i = VALUE + 1;
    printf( "%d\n", i );
    return 0;
}
```

f)

```
// fuehre Schleife 12x aus
for(int i = 0; i < 13; ++i ) {
    //..
}
```

g)

```c
#include <stdio.h>
// Eine Zahl umstaendlich verdoppeln
int main() {
    int zahl = 1;
    switch (zahl) {
        case 1:
            zahl = zahl + 1;
        case 2:
            zahl = zahl + 2;
        case 3:
            zahl = zahl + 3;
    }
    printf("Die doppelte Zahl lautet %d\n",zahl);
    return 0;
}
```

h)

```c
#include <stdio.h>
// Eine Zahl umstaendlich verdoppeln
int main() {
    int zahl = 1;
    if (zahl == 1)
    {
        zahl = zahl + 1;
    } if (zahl == 2)
    {
        zahl = zahl + 2;
    } if (zahl == 3)
    {
        zahl = zahl + 3;
    }
    printf("Die doppelte Zahl lautet %d\n",zahl);
    return 0;
}
```

i)

```c
#include <stdio.h>

int main() {
    int dividend = 6;
    int divisor = 4;
    float result = dividend / divisor;
    printf("Das Ergebnis lautet %f\n",result);
    return 0;
}
```