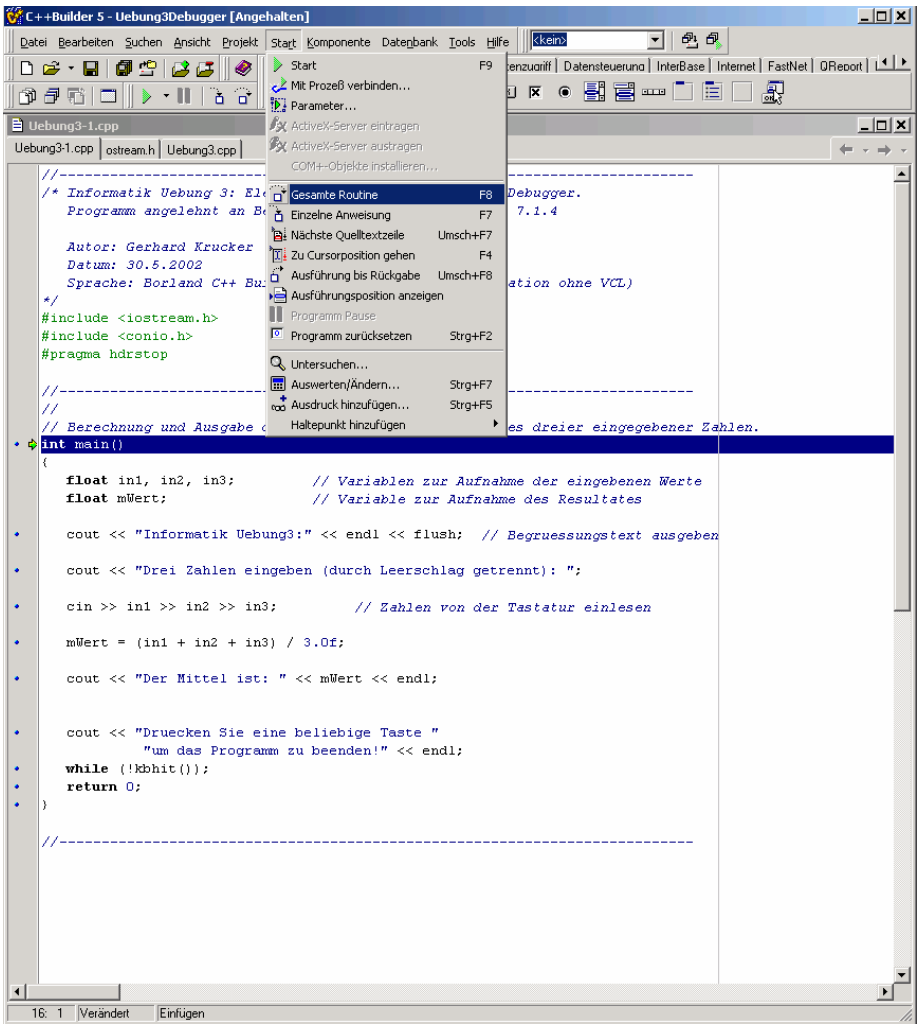
U3 Debugger I

Der Debugger ist ein universelles Test- und Fehlersuchwerkzeug für Programme. Es ermöglicht den Programmcode kontrolliert abzuarbeiten, indem Haltepunkte im Ablauf gesetzt werden, kann der Code schrittweise durchlaufen werden. Ebenso können Werte von Variablen betrachtet und geändert werden.
Wir beschränken uns in einer ersten Einführung auf die Möglichkeiten der schrittweisen Programmausführung, bei der Instruktion um Instruktion abgearbeitet wird und der Erfolg sofort sichtbar wird.
Bedingung für das Debugging ist aber mindestens ein startbares EXE-File mit Debuggerinformation. Der Debugger ist also nicht geeignet, um syntaktische Fehler zu behandeln.

Einzelschrittmodus

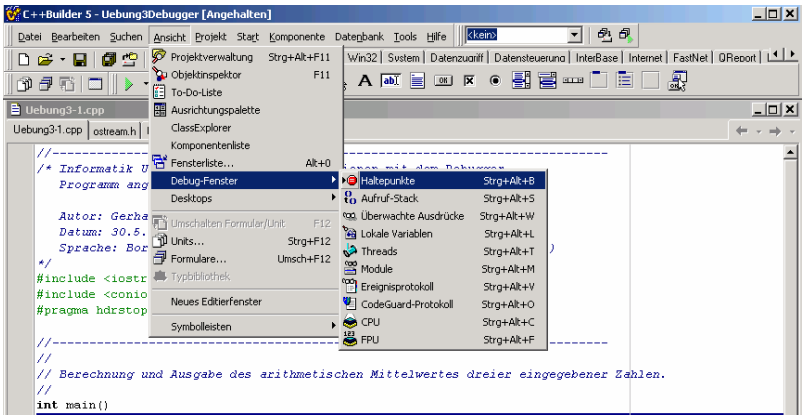
Wurde das Programm kompiliert und gelinkt, d.h. ein lauffähiges EXE-File erstellt, kann über den Menüpunkt *Start/Einzelne Anweisung* (oder F7) das Programm gestartet werden. Es erscheint ein blauer Balken, der die nächste auszuführende Instruktionszeile markiert, hier bei `main(..)`.



Weitere Schritte können mit F7 oder F8 durchgeführt werden. Die Unterschiede beider Aktionen liegen in der Behandlung von Funktionsaufrufen:

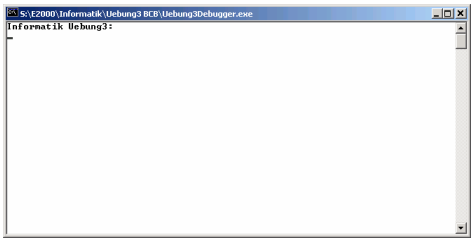
F8: Ein Funktionsaufruf wird als einzelne Instruktion behandelt.
F7: Bei einem Funktionsaufruf wird in die Funktion verzweigt und die Funktion Zeilenweise behandelt.

Die unterschiedlichen Debug-Aktionen werden in eigenen Fenstern dargestellt. Sie werden über das Menu *Ansicht/Debug-Fenster* ausgewählt:

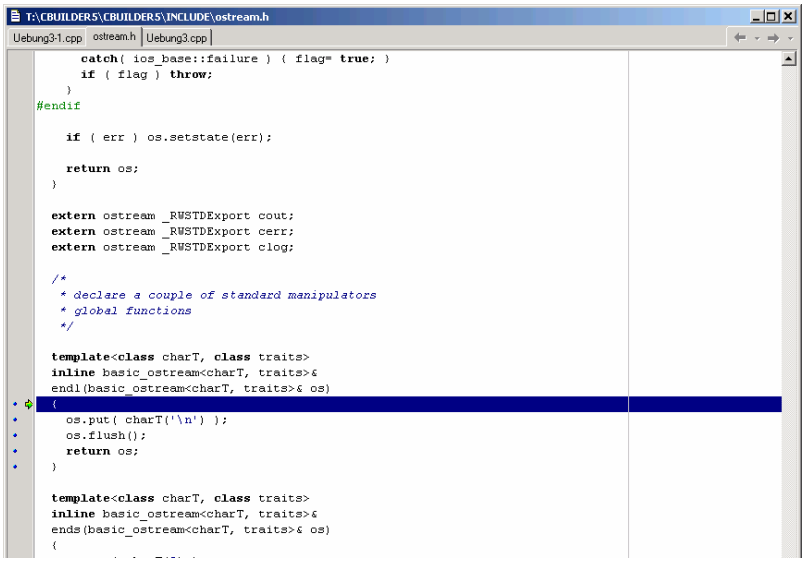


Meist wird man das Fester für lokale Variablen und überwachte Ausdrücke wählen.

Wird nun 2x F8 gedrückt wird die cout - Funktion mit dem Begrüssungstext als einzelne Instruktion abgearbeitet. Die Ausgabe erfolgt im Output-Fenster:

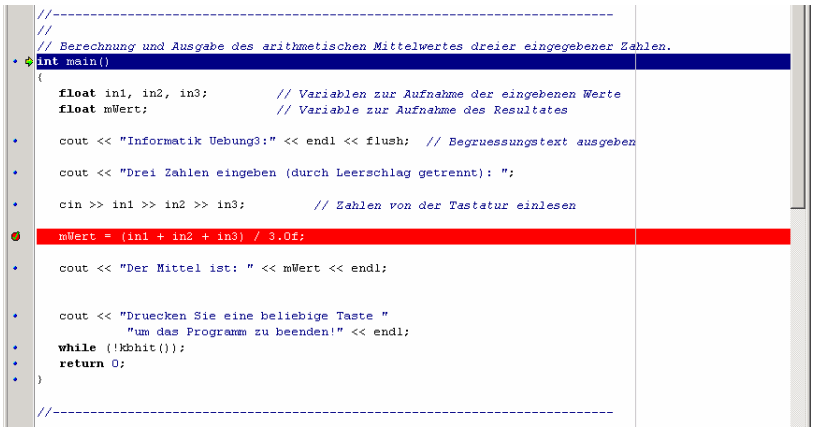


Je nach Konfiguration des Compilers könnte auch die cout -Funktion bis zu einem gewissen Grad schrittweise durchlaufen werden, was aber kaum interessante Erkenntnisse bringt.



Haltepunkte

Weiter können Haltepunkte (Breakpoints) im Code gesetzt werden. Beim Erreichen eines solchen Haltepunktes wird der normale Programmlauf unterbrochen. Dazu setzt man von dem Programmstart den Cursor auf die Zeile, wo ein Programmhalt verlangt wird. Dann klickt man auf den blauen Punkt links der Codezeile. Die gesamte Zeile wird nun rot markiert. Erneutes Doppelklicken löscht den Haltepunkt wieder.



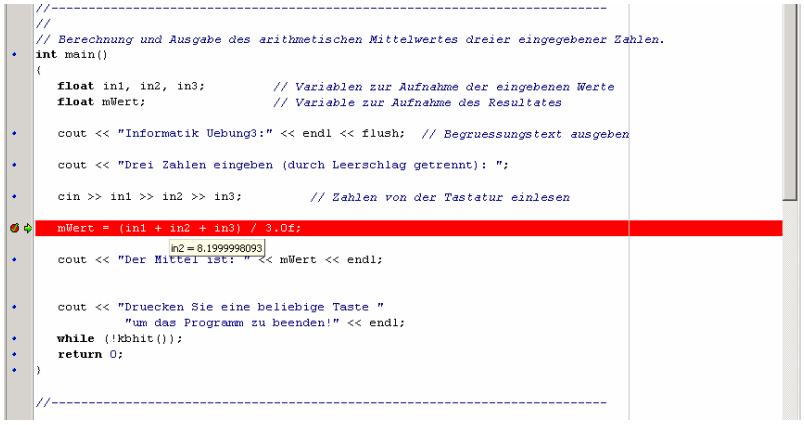
```
//---------------------------------------------------------------------------
//
// Berechnung und Ausgabe des arithmetischen Mittelwertes dreier eingegebener Zahlen.
int main()
{
   float in1, in2, in3;         // Variablen zur Aufnahme der eingebenen Werte
   float mWert;                 // Variable zur Aufnahme des Resultates

   cout << "Informatik Uebung3:" << endl << flush;  // Begruessungstext ausgeben

   cout << "Drei Zahlen eingeben (durch Leerschlag getrennt): ";

   cin >> in1 >> in2 >> in3;         // Zahlen von der Tastatur einlesen

   mWert = (in1 + in2 + in3) / 3.0f;

   cout << "Der Mittel ist: " << mWert << endl;


   cout << "Druecken Sie eine beliebige Taste "
           "um das Programm zu beenden!" << endl;
   while (!kbhit());
   return 0;
}

//---------------------------------------------------------------------------
```

Wird das Programm gestartet, erfolgt der Programmlauf bis zum Erreichen nächsten Haltpunktes. Bei umfangreichen Programmen kann so selektiv ein Programmlauf bis zum zu untersuchenden Teil erfolgen, ohne dass im Einzelschrittmodus gearbeitet werden muss.

Varblenwerte untersuchen

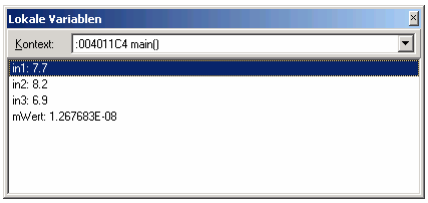
Ist das Programm im Haltemodus, können alle zur Zeit sichtbaren Variablenwerte in ihrem Wert betrachtet und wenn notwendig auch geändert werden.

Am einfachsten erfolgt die Betrachtung, indem man mit dem Cursor auf die Variable fährt. Nach ca. 1 Sekunde geht ein Tooltipp-Fenster auf, welches den aktuellen Wert anzeigt.

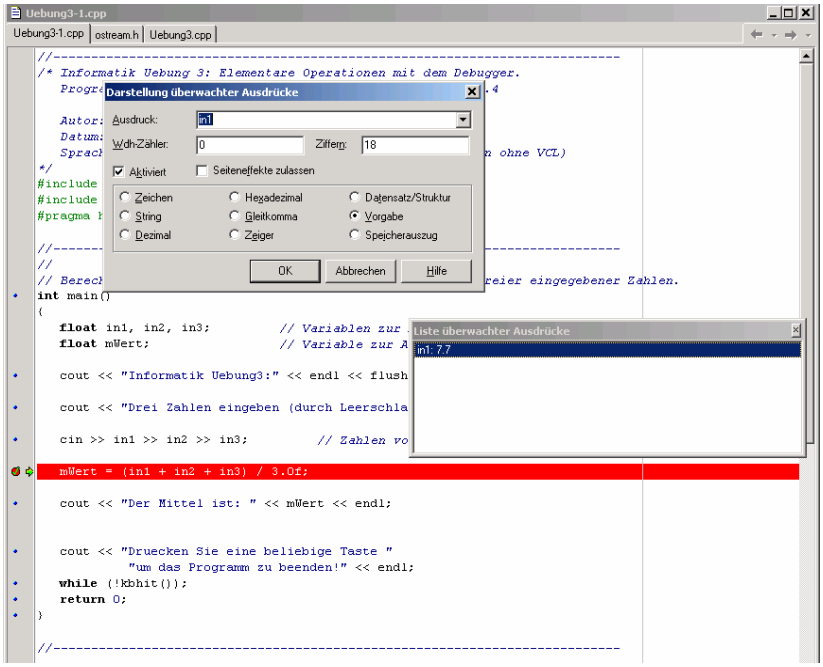


```
//---------------------------------------------------------------------------
//
// Berechnung und Ausgabe des arithmetischen Mittelwertes dreier eingegebener Zahlen.
int main()
{
   float in1, in2, in3;         // Variablen zur Aufnahme der eingebenen Werte
   float mWert;                 // Variable zur Aufnahme des Resultates

   cout << "Informatik Uebung3:" << endl << flush;  // Begruessungstext ausgeben

   cout << "Drei Zahlen eingeben (durch Leerschlag getrennt): ";

   cin >> in1 >> in2 >> in3;         // Zahlen von der Tastatur einlesen

   mWert = (in1 + in2 + in3) / 3.0f;
                 in2 = 8.1999998093
   cout << "Der Mittel ist: " << mWert << endl;


   cout << "Druecken Sie eine beliebige Taste "
           "um das Programm zu beenden!" << endl;
   while (!kbhit());
   return 0;
}

//---------------------------------------------------------------------------
```

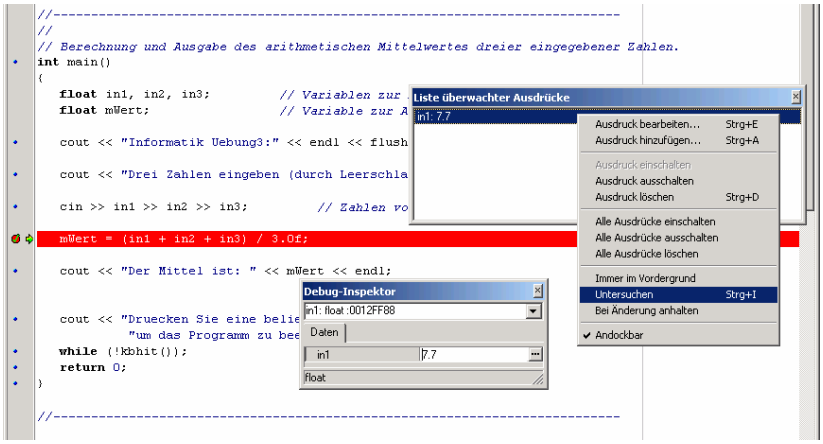
Weiter kann auch das Debug-Fenster für die lokalen Variablen über *Ansicht/Debug-Fenster/Lokale Variablen* aktiviert werden. Es zeigt dann immer sämtlich sichtbaren lokalen Variablen an:



Weiter kann auch mit sog. Watch-Listen gearbeitet werden. Sie beihalten die zu überwachenden Ausdrücke. Sie werden durch Doppelklicken auf das Fenster mit den zu überwachenden Ausdrücken mittels Dialog eingegeben und werden fortan angezeigt.



Ein Wert kann in der Liste der überwachten Ausdrücke über ein Klick mit der rechten Taste und *Untersuchen* geändert werden:



Hinweis

Folgens Phänomen wurde beobachtet:
Werden `/*  */`-Kommentare vor der ersten Zeile mit dem `//- - - - - - - - - - - - - - - - - -` Kommentar eingefügt, werden beim Debuggen die Zeilen nicht korrekt angezeigt (Versatz um die Grösse des Kommentares).

Aufgaben

1.Implementieren Sie nach vorliegendem Beispiel und Lehrbuch Kap. 7.1.3 das Programm und testen Sie es mit dem Debugger aus.

2. Setzen Sie Haltpunkte und untersuchen / ändern Sie die lokalen Variablenwerte.

3. Kapitel 6 des Lehrbuches repetieren und Aufgaben in Kap. 6.6 lösen.