



Wanzen weg vom Fenster

Wer bereits selbst eine Anwendung geschrieben hat, weiß, wie hartnäckig Fehler sein können. Microsoft hat im Visual C++ 6 einen starken Debugger dagegen eingebaut.



# Debugging unter Windows

WOLFGANG SOLTENDICK

**H**eute sind Editor, Compiler und Debugger zu einem Programmpaket verschmolzen, so dass auf sehr effektive Weise eine Anwendung entwickelt werden kann. Dies gilt für das aktuelle Visual-C++-6-Paket zur Entwicklung von Programmen von Microsoft.

## QUICK INDEX

- ▶ **Voraussetzungen zur Untersuchung**  
Bevor Sie eine Anwendung untersuchen können, müssen Sie einiges beachten und einstellen.
- ▶ **Untersuchung der Anwendung**  
Bei der Untersuchung eines Programms führen verschiedene Wege zum Ziel.
- ▶ **Haltepunkte**  
Anhalten und weitermachen  
Breakpoints und Einsprungmethoden
- ▶ **Nützliche Fenster während der Untersuchung**  
Für die Untersuchung des Programms mit Hilfe des Quelltextes ist es wichtig, sich die Inhalte von Variablen, Objekten usw. anschauen zu können.
- ▶ **Das assert-Kommando**  
Zur Laufzeit Fehler finden

## Voraussetzungen zur Untersuchung

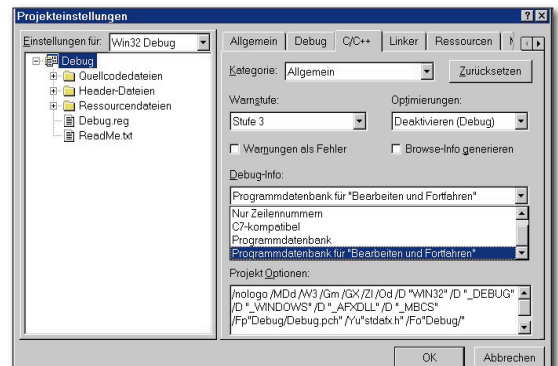
Standardmäßig erzeugt der VC6-Kompiler eine Release- und eine Debug-Version eines Projektes. Erstere bezeichnet die endgültige und auslieferbare Version und letztere ist zum Testen des Codes gedacht. Der Unterschied zwischen den beiden Versionen besteht darin, dass jeweils unterschiedliche Kompiler-Einstellungen verwendet werden.

In der Debugversion werden die kompletten symbolischen Debugging-Informationen im Microsoft-Format eingefügt und es wird nicht optimiert. In der Release-Version wird optimiert und es werden keine symbolischen Debugging-Informationen erzeugt.

Natürlich steht es Ihnen frei, die Standardeinstellungen in den beiden Versionen Ihren Bedürfnissen anzupassen. Alle diese Optionen finden Sie in einem Dialog, den Sie über den Menüpunkt "Projekt/Einstellungen" öffnen können.

Die wichtigsten Optionen finden Sie auf der Seite "C/C++". Achtung, die Optimierung müssen Sie von Hand abschalten. Sonst würde der Kompiler bestimmte Befehle verschieben und organisiert damit die Ablauf-

struktur neu. Das entspricht dann nicht mehr dem Ablauf in Ihrem Quelltext, so dass der Debugger nicht mehr die zugehörige Quelltextzeile identifizieren und anzeigen kann. Damit die Debugging-Einstellungen wirksam werden können, muss das Symbol "\_DEBUG" definiert sein. Dazu können Sie die Kommandozeilenoption "/D\_DEBUG" angeben. Spezielle Programmteile, die nur in der Debug-Version wirksam werden sollen, können Sie in ein "#ifdef \_DEBUG ... #endif"-Konstrukt einfügen. Außerdem müssen Sie für die Untersuchung die Debugging-Versionen der MFC-Bibliotheken verwenden. Um nicht jedes Mal alles von Hand einzustellen, ist es notwendig, als aktives Projekt die Debug-Version auszu-



DIE OPTIONEN ZUM Einstellen des Kompilers.



wählen. Dieses erledigen Sie über den Menüpunkt "Erstellen/Aktive Konfiguration festlegen..."

## ■ Untersuchung der Anwendung

Nachdem Sie die Grundvoraussetzungen geschaffen haben, können Sie nun daran gehen, Ihre Anwendung genauer zu untersuchen. Dazu wählen Sie den



**DAS FENSTER** mit den Debug-Werkzeugen.

Menüpunkt "Erstellen/Debug starten/Ausführen" aus. Allerdings sehen Sie beim ersten Starten die Meldung, dass die Quelltexte veraltet sind (die Debug-Informationen müssen aktualisiert werden) und neu übersetzt werden müssen. Sie läuft nach dem Start normal und hält erst an, wenn ein Haltepunkt erreicht ist.

In der Regel wollen Sie eine Anwendung Schritt für Schritt ausführen, wofür der Menüpunkt "Erstellen/Debug starten/In Aufruf springen" vorgesehen ist.

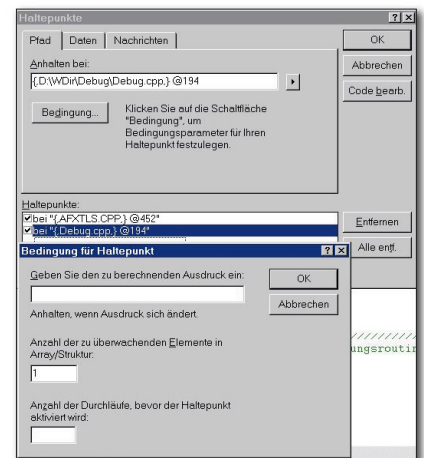
Nach dem Klick darauf wird das entsprechende Quelltextfenster geöffnet und Sie sehen in der Zeile, die als nächste abgearbeitet werden soll, am linken Fensterrand einen gelben Pfeil. Außerdem wird Ihnen das "Debug"-Fenster

angezeigt, das alle Werkzeuge zur Verfügung stellt, die zum Untersuchen einer Anwendung nützlich sein können.

Zu den meisten Button im Fenster "Debug" existieren entsprechende Einträge im Debug-Menü. Mit Hilfe des Buttons "In Aufruf springen" (oder der Taste [F11]) können Sie Zeile für Zeile weiter durchgehen. Allerdings geraten Sie dabei in die Tiefen der Debug-Version der MFC-Bibliothek, was nicht sehr erquickend ist. Außerdem sieht der Bildschirm nach einiger Zeit sehr interessant aus, da viele Fenster geöffnet worden sind, um die Quelltexte anzuzeigen.

Der größte Schwachpunkt dieser Vorgehensweise ist, dass Sie nach geraumer Zeit zur Nachrichtenschleife kommen, die natürlich auch in MFC-Programmen existiert. Sie haben aber die Möglichkeit, den Cursor in die Zeilen Ihres Programms zu setzen, die Sie interessieren und das Programm bis dahin laufen zu lassen, wo es automatisch stoppt. Meistens setzt man einen Haltepunkt, an dem angehalten wird, sobald man daran vorbeikommt.

Befinden Sie sich in der gewünschten Zeile, können Sie von dort die Untersuchung Ihres Programms so fortsetzen, wie Sie sie zuvor begonnen haben. Sie haben die Möglichkeit, Zeile für Zeile zu untersuchen ("In Aufruf springen") oder die aktuelle Funktion komplett ausführen zu lassen, bis Sie zum Rücksprung kommen ("Ausführen bis Rücksprung"). Gelangen Sie bei der Einzel-



**DAS SETZEN EINER Bedingung** für einen Haltepunkt

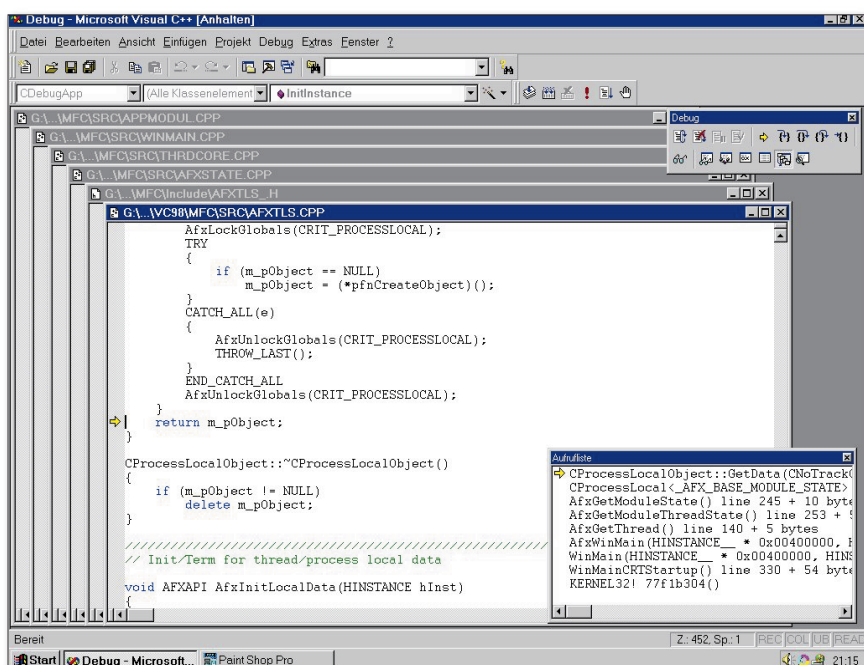
ausführung an einen Funktionsaufruf, haben Sie wieder die Wahl, in die Funktion zu springen ("In Aufruf springen"), oder sie in einem Stück ausführen zu lassen ("Aufruf als ein Schritt"). Daneben können Sie das Programm alternativ wieder bis zu der Stelle ausführen, an der sich der Cursor aktuell befindet ("Ausführen bis Cursor"). Beachten Sie, dass diese Debug-Funktion keine Wirkung zeigt, wenn sich der Cursor in der Zeile befindet, die als nächstes ausgeführt werden soll.

Mit diesen Funktionen haben Sie bereits das Grundgerüst kennen gelernt, um Ihr Programm untersuchen zu können. Dabei ist es ratsam, je nach Untersuchungsfall eine Kombination der zuvor beschriebenen Möglichkeiten anzuwenden.

## Haltepunkte

Einen Haltepunkt setzen Sie am einfachsten, wenn Sie in die zugehörige Zeile im Quelltext gehen und im lokalen Menü den Punkt "Haltepunkt einfügen/löschen" auswählen. Dadurch sehen Sie links vom Quelltext ein rotes Feld, das den Haltepunkt repräsentiert.

Eine Übersicht über die gesetzten Haltepunkte erreichen Sie über das Fenster "Haltepunkte" ("Bearbeiten/Haltepunkte..." oder [Alt]+[F9]). Geben Sie keine Bedingung an, wird die Ausführung jedes Mal in der Zeile angehalten, in der sich der Haltepunkt befindet. Haben Sie einen Fehler beseitigt, kön-



**DIE GEÖFFNETEN** Quelltextfenster. Es können sehr viel werden, manchmal sogar zuviele.

## BÜCHER

Redaktion Toolbox, Visual C++ 6, C&L-Verlag, 1999, ISBN 3-932311-20-5



Name	Wert
m_msgCur	{msg=0x0000036a wp=0x00000000 lp=0x00000000}
hwnd	0x000403b8
message	0x0000036a
wParam	0x00000000
lParam	0x00000000
time	0x012ade54
pt	{x=0x0000018a y=0x0000023d}
x	0x0000018a
y	0x0000023d

DIE ANZEIGE einer überwachten Struktur im Hauptspeicher.

nen Sie übrigens einen Haltepunkt wieder ausschalten (im lokalen Menü: "Haltepunkt deaktivieren").

## Nützliche Fenster während der Untersuchung

Bei der Untersuchung eines Programms kommt es häufig vor, dass man sich einen Überblick über die Werte von Variablen verschaffen möchte. Dies funktioniert am bequemsten im Überwachungsfenster ("Ansicht/Debug-Fenster/Überwachung"), das Ihnen meist am unteren Bildrand ("Gedockte Ansicht") angezeigt wird.

Alternativ haben Sie die Möglichkeit, sich die Variablen in einem Fenster ausgeben zu lassen, die nur zur aktuellen Funktion gehören ("Ansicht/Debug-Fenster/Variablen"). Damit können Sie sich einen schnellen Überblick verschaffen und müssen die Namen nicht immer im Fenster "Überwachung" eingeben. Auf der Seite "Auto" finden Sie Informationen über die Variablen, die in der aktuellen und in der vorigen Anweisung bzw. Zeile verwendet werden. Die Seite "Lokal" enthält die Daten der Variablen, die in der kompletten, aktu-

ellen Funktion Verwendung finden und unter "this" sehen Sie die Darstellung des Objektes, auf das der Zeiger verweist.

In allen zuvor beschriebenen Fenstern können Sie die Werte der angezeigten Variablen direkt ändern. Der neue Wert wird bei der weiteren Abarbeitung des Programms verwendet, so dass Sie damit Berechnungsfehler oder ähnliches korrigieren können. Untersuchen Sie Ihr Programm genauer, können Sie sich den Inhalt der Prozessorregister (Button "Register" im Fenster "Debug") ausgeben lassen. Das kann mitunter nützlich sein, wenn Sie in der endgültigen Version versuchen, einen Fehler aufzuspüren, der durch die Optimierung zustande kommt (was eigentlich sehr selten auftritt).

Außerdem haben Sie die Möglichkeit, sich ausgeben zu lassen, welche Funktionen bisher aufgerufen worden sind, um zur aktuellen Position zu gelangen ("Ansicht/Debug-Fenster/Aufrufliste").

## Das ASSERT-Kommando

Wenn Sie sich die Quelltexte näher ansehen (speziell die der MFC-Bibliothek), werden Sie häufig auf den Befehl "ASSERT" stoßen. Mit diesem steht ei-

ne weitere Möglichkeit zur Untersuchung eines Programms zur Verfügung, wobei die in den Klammern angegebene Bedingung zur Laufzeit der Anwendung ausgewertet wird. Dies passiert nicht in der Release-Anwendung des Programms, so dass diejenigen Leute, die damit arbeiten müssen, nichts davon bemerken. In der Release-Version wird ein "ASSERT"-Befehl so behandelt, als wäre er nicht vorhanden, genauer: "ASSERT" ist nur dann wirksam, wenn das Symbol "\_DEBUG" definiert ist.

Trifft die angegebene Bedingung zu, läuft das Programm normal weiter! Im anderen Fall erscheint eine Dialogbox auf dem Bildschirm, die angibt, dass ein Fehler aufgetreten ist.

Damit haben Sie eine Möglichkeit, den Wert einer Variablen zur Laufzeit zu überprüfen und gegebenenfalls das Programm anzuhalten. Außerdem werden Ihnen die Zeile und das Quelltext-Modul angezeigt, in welcher der Fehler aufgetreten ist. Beachten Sie bei der Angabe einer Bedingung, dass Sie den zu überprüfenden Wert nicht noch selbst verändern. Zeilen wie

ASSERT (i++ > 0);

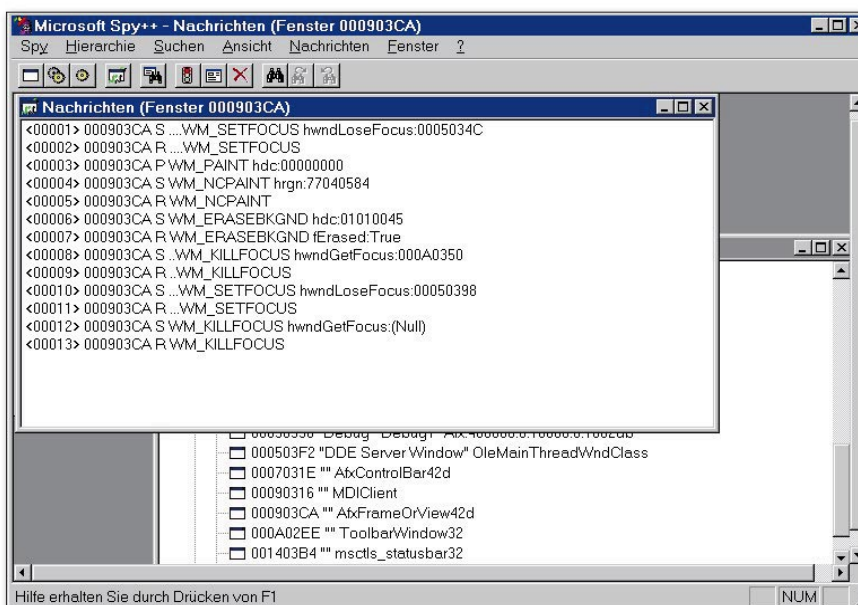
müssen Sie auf jeden Fall vermeiden! Diese hat zudem noch den Nachteil, dass sich die Debug- und die Release-Version in Ihrem Verhalten unterscheiden, da die Erhöhung der Variablen "i" nur in der Debug-Version ausgeführt wird.

## Weitere Werkzeuge

Zusätzlich zu den vorgestellten Werkzeugen existieren eine Reihe weiterer, externer Programme oder Skripte, mit deren Hilfe Sie Informationen über ein Windows-System bestimmen können. Alle diese finden Sie im "Extras"-Menü von VC6.

Dazu gehört das Programm "Spy" ("Extras/Spy++"), mit denen Sie bestimmen können, welche Nachrichten an eine Anwendung gesendet werden. Damit können Sie überprüfen, ob eine Nachricht auch tatsächlich an das zu testende Programm gesendet wird oder nicht. Aber erschrecken Sie nicht, wenn Ihnen viele Nachrichten nicht bekannt vorkommen sollten, es sind 849 vordefiniert und benutzerdefinierte sind ebenfalls möglich.

Damit haben Sie einen Überblick über die Möglichkeiten erhalten, die Ihnen VC6 bietet. Allerdings existieren noch wesentlich mehr Funktionen, obwohl Sie einen Großteil davon nur in Spezialfällen benötigen.



ANZEIGE DER NACHRICHTEN an die im Debugger laufende Anwendung