Courses 2DCx, C4: File, FAQs

Was ist ein dynamischer Array ?

Ein statischer Array (z.B. `int FELD[100];`) hat eine feste Länge über seine gesamte Lebensdauer. Man kann nichts anhängen, wenn er sich als zu klein herausstellt, man kann ihn nur insgesamt freigeben, aber keine Teile davon, wenn er sich als zu groß herausstellt. Vorteil: Er benötigt zur Laufzeit keinerlei inneren Verwaltungsaufwand.
Ein dynamischer Array (z.B. `std::vector< int > FELD;`) hat undefinierte Länge. Mit Befehlen wie `FELD.push_back(element)` kann man ihn beliebig verlängern (und mit anderen Befehlen auch kürzen). Trotzdem kann man sicher sein, dass die Elemente im Speicher logisch lückenlos in aufeinanderfolgenden Speicherplätzen liegen. Das spart Speicherplatz, weil man keine Platzreserven anlegen muss und macht den Code elegant. Nachteil: Es ensteht unsichtbarer interner Verwaltungsaufwand zur Laufzeit (z.B. bei Platzmangel muss alles im RAM auf eine andere Stelle mit mehr Platz umkopiert werden).

Was ist STL ?

Die Standard Template Library ist eine Bibliothek von Containerklassen, die als fester Bestandteil von C++ in die Sprache adoptiert wurde.
Container sind Arrays, Listen, Bäume etc. Obwohl die Datentypen der Containerelemente beliebig kompliziert frei vereinbar sind, gibt es standardisierte Musterelemente. Hält man sich an diese Muster = Templates, dann kann man standardisierte Zugriffsoperatoren = Iteratoren und Algorithmen benutzen. Es gibt 5 Typen von Iteratoren: Random Access, bidirectional, Forward, Input, Output und es gibt standardisierte Algorithmen: count, copy, replace, reverse, sort etc.

Warum STL anstatt MFC-CArray ?

Die MFC sind um einige Jahre älter als die STL. Als es STL noch nicht gab, waren die MFC-Entwickler gezwungen, eigene Implementierungen dynamischer Arrays, verketteter Listen und Bäume in einer eigenen Klassenfamilien zu bieten: die sogenannten MFC-Sammelklassen `CArray, CList, CMap`, die jeweils 5 bis 10 Unterklassen für gängige Datentypen enthalten und alle mit `#include < afxtempl.h >` deklariert sind.
Inzwischen sind diese MFC-Sammelklassen überflüssig geworden dadurch, dass die moderne STL zum integralen Teil der Sprache C++ erhoben wurde. Microsoft kann die alten Klassen aber nicht streichen, sie werden in alle Ewigkeit in den MFC mumifizieren, denn grosse Mengen von MFC-basierten Applikationen bleiben darauf angewiesen.
Heute gibt es nur noch zwei Argumente die für die MFC-Sammelklassen und gegen die modernere STL sprechen:
1. MFC-Arrays sind etwas schneller als STL-Vektoren, letztere sind dafür funktionell mächtiger.
2. MFC-Arrays sind in der MFC-Bibliothek enthalten, für die STL-Arrays wird die STL-Bibliothek zusätzlich eingebunden. Das bläht alle EXEs um einige kB auf.

Was ist Document-View-Architektur ?

Fast alle Programme enthalten Data Processing = Umschaufeln von Daten von Platte ins RAM, innerhalb des RAM und vom RAM auf Platte. Moderne Programme enthalten zusätzlich Computer Graphics = Visualisierung von Daten durch Bildchen (auch Buchstaben sind Bildchen!) in einem Client-Fenster (Single Document Interface SDI) oder mehreren Clientfenstern (Multiple Document Interface MDI) auf dem Monitor.
Die Trennung der Programmwelten Data Processing und Computer Graphics in die beiden Klassen CDocument und CView ist modern und hygienisch, wird allerdings mit deutlichem Verwaltungsaufwand erkauft. Der ApplicationWizard nimmt einem die Vorarbeiten ab (Check-Box: `Unterstützung der Dokument-/Ansicht-Architektur`).
Die zusätzliche Arbeit beim Programmieren in `CView` besteht darin, dass man für die Datenzugriffe dauernd einen Pointer namens `pDoc->` auf `CDocument` braucht.
Beispiel `void CFile1View::OnDraw(CDC* pDC)` in der Musterlösung:

```
void CFile1View::OnDraw(CDC* pDC)
{
1  CFile1Doc* pDoc = GetDocument();
2  ASSERT_VALID(pDoc);
3  pDC->TextOut( 10, 10, "zeichnen->speichern->schliessen->laden" );
4  int n = pDoc->p.size();
5  if ( n < 2 ) return;
6  pDC->Polyline( pDoc->p.begin(), n);
}
```

Zeilen 1 und 2 sind kostenlos schon eingetragen.
Zeile 1 besorgt einen Pointer `pDoc` auf die `CDocument`-Klasse, die in der Musterlösung `CFile1Doc` heißt.
Zeile 2 ist unnötig. Sie generiert im `Debug-Mode` eine Fehlermeldung, wenn es keine `CDocument`-Klasse gibt.
Zeile 3: `pDC->` ist der Pointer auf den DC (siehe Parameter im Funktionskopf) und hat nichts mit der Architektur und `CFile1Doc` zu tun.
Zeile 4: `pDoc->p.size()` liefert die aktuelle Zahl der Elemente im dynamischen Array `p`, der in `CFile1Doc` steht.
Zeile 5: uninteressant
Zeile 6: `pDoc->p.begin()` meint die Adresse des ersten Elements in dem dynamischen Array `p`, der in `CFile1Doc` steht.

Faustregeln, wo man am besten welche Variablen unterbringt:
1. Man verwalte alle Variablen etc., die mit der Harddisk zu tun haben, grundsätzlich und immer `public:` in `CDocument`.
2. Von den restlichen Variablen etc., verwalte man die, die mit dem Bildschirm zu tun haben `private:` in `CView`.
3. Die jetzt noch verbleibenden Variablen etc. verwalte man lokal in den Einzelfunktionen.

Was ist Serialisierung ?

Die Programmierung der Menübefehle `Datei Öffnen...` und `Datei Speichern unter...` ist sensationell einfach unter MFC. Das verdanken wir der MFC-Klasse `CArchive`, hinter der sich die Klasse `CFile` versteckt. `CArchive` überlädt die Ein- und Ausgabeoperatoren << und >> und Read und Write so konsequent, dass der enorm komplizierte Vorgang sich als eine einfache Zuweisung formulieren läßt: z.B. `ArchiveObjekt << IrgendeinObjektAufHardisk;`.
Dieses Wunder konsequenter OOP-Vererbung nennt Microsoft "Serialisierung", was andeuten soll, dass ein Festplattenzugriff zwar physikalisch unglaublich kompliziert ist, aber logisch sich am besten genauso verhalten soll wie ein Zugriff auf einen dynamischen Array im RAM.

Jump to: Kurzartikel über CArchive aus dem MSDN

Wozu braucht man Polygone im WEB ?

Die gängigen Browser kennen für Graphiken nur die Rasterformate BMP, GIF, JPEG und PNG.
Wenn Sie Ihre Zeichnungen von draw1 ins WEB stellen wollen, dann geht das nur in Form einer Pixelkopie der ClientArea von draw1 (die Kopie natürlich in Form einer BMP, GIF etc.-Datei), aber nicht in Form eines Polygons.
Diese Pixelkopie hätte katastrophale Eigenschaften:
1. Sie benötigt mindestens 1000x mehr Speicherplatz und Übertragungszeit verglichen mit unserem serialisierten Polygon.
2. Man kann sie weder zoomen noch rotieren, es sei denn mit Gewalt und unter hohem Qualitätsverlust.
3. Die Kopie kann man nicht ins Dokument einbinden, sondern man muss sie als eigenes File verwalten, transportieren und referenzieren.
Vektorformate kennen diese Probleme nicht. Sie müssen allerdings zur Laufzeit in Rasterformate umgewandelt werden = Rendering.
Zusammenfassung:
Rasterformate eignen sich nur für natürliche Bilder (Photo, Video). Sie brauchen zwar keine schnellen Rechner, aber eine schnelle Verbindung zum Internet.
Vektorformate eignen sich nur für künstliche Bilder (Graphic, Comic, Game). Sie brauchen zwar schnelle Rechner, aber keine schnelle Verbindung zum Internet.
Macromedia Flash ist das einzige Vektorformat, das im WEB weite Verbreitung gefunden hat. Der WEB-Entwickler benötigt einen teueren Editor (=Macromedia Flash) und jeder Konsument muß ein Plugin akzeptieren.
Es gibt noch weitere Vektorformate, die aber weniger Verbreitung gefunden haben:
- VRML = Virtual Reality Modeling Language, standardisiert vom W3C
- Pulse3D, Cult3D, SCOL und Shockwave3D.
Allen diesen Vektorformaten ist gemeinsam, daß sie nicht nur Polygone speichern, sondern auch Anweisungen enthalten, wie die Polygone zu zoomen, zu rotieren, wie die vom Polygon eingeschlossenen Flächen zu füllen und zu schattieren sind. Sie sind also nicht nur Speicherform sondern sie enthalten auch alle Anweisungen für Animation und Rendering.
Sie sind Zwitter aus Polygonformaten und Interpretersprachen.
Mit Hilfe von Plugins laden die Browser die Polygone ein einziges Mal am Anfang der Darstellungskette und interpretieren dann (bei Animationen mehrfach) die Anweisungen zum Zoomen, Rotieren und Rendern.
Einige intelligente Plugins passen dabei automatisch die Anzahl der Polygonecken an die Rechenleistung von CPU und Graphikkarte ihres Zielrechners an (MRM = Multiresolution Mesh), indem sie die Polygone automatisch vergröbern durch weglassen von Vertices oder verfeinern durch interpolieren zusätzlicher Vertices (=Subdivision Surfaces), je nach Rechenleistung des Zielrechners und dessen Graphikkarte.

Jump to: http://developer.intel.com/ial/3dsoftware/

Was ist SVG, was ist VML ?

Es gibt 2 neue XML-basierte Vektorformate die Zukunftspotential haben:
1. SVG = Scalable Vector Graphics vom W3Cstandardisiert. Sehr neu und noch kaum irgendwo implementiert. Bisher einziger Editor ist Adobe Illustrator 9.0 und die Browser müssen ein spezielles Plugin von Adobe installieren.
2. VML = Vector Markup Language von Macromedia und Microsoft. VML ist älter und Vorbild von SVG, wobei beide Sprachen ziemlich ähnlich sind. VML-Code kann man erzeugen mit allen MS-Office- und VISIO-Applikationen. Es wird ohne jedes Plugin vom MS-InternetExplorer ab Version 5.0 (aber nicht von Netscape) interpretiert.

SVG und VML sind von XML abgeleitete Sprachschemata. Das bedeutet, dass sämtliche SVG+VML-Sprachkonstrukte in gültiger XML-Syntax vorliegen und von jedem XML-Parser als zulässige Eingaben angesehen werden. SVG+VML-Tags kann man in HTML- und in XML-Dateien einbetten. Mit Cascading Style Sheets CSS kann man Objektattribute, Größe und Position festgelegen. In VML formulierte Objekte fügen sich ins Document Object Model DOM organisch ein, d.h. sie können sich auf Änderung der Dokumente automatisch mitverändern.
VML-Graphiken beginnen mit `< html xmlns:v="urn:schemas-microsoft-com:vml" >`.
Der Parameter `xmlns:v="urn:schemas-microsoft-com:vml"` spezifiziert einen sogenannten XML-Namespace und gibt an, dass alle Tags, denen ein `v:` vorangestellt ist, als VML-Tags zu interpretieren sind, also zur VML-Syntax gehören.
Mit der zweiten wichtigen Zeile `< style &gtv\:* {behavior:url(#default#VML);}< /style >` werden alle `v:`-Befehle an die VML-Engine des InternetExplorer gelenkt. Diese VML-Engine = Rendering Engine setzt zur Laufzeit die VML-Anweisungen in Windows-GDI-Aufrufe um.

In Musterprogramm FirstFile = file1 wird ein vordefiniertes VML-Objekt, nämlich `v:polyline` benutzt. Dessen Attribute `strokecolor, strokeweight, points` sind in CSS-Stylesheets gekapselt.
Weitere vordefinierte (in FirstFile = file1 nicht verwendete) Objekte sind `v:line, v:rect, v:oval, v:shape.`

Links

C++, MFC:
http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/visualc/vcedit/vcstartpage.htm

STL:
www.sgi.com/Technology/STL/
www.cs.rpi.edu/projects/STL/htdocs/stl.html
www.xraylith.wisc.edu/~khan/software/stl/STL.newbie.html

SVG:
(1) www.w3.org/Graphics/SVG/
(2) www.w3.org/TR/WD-SVG
(3) www.w3.org/Graphics/SVG/SVG-Implementations

VML:
mit Beta-Graphik-Editor für VML (900 kB):
(1) msdn.microsoft.com/standards/vml/ref/default.asp
(2) www.w3.org/TR/NOTE-VML