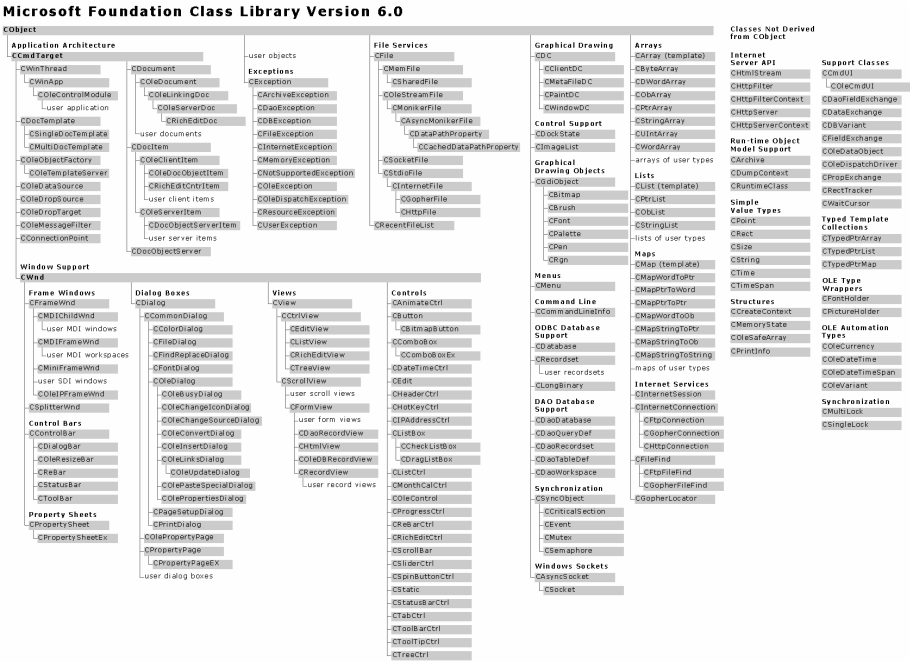
Grundlagen in Visual Studio
MFC und .net

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt



Hilfsklassen in der MFC

- **CString**
- **Cfile**
- **CSerial**
- **CCompressedFile**
- **CRegistry**

Klasse

Eigenschaften:

- Speichert einen Zero-String
- Copy-Konstruktor
- Zuweisung-Operator
- Methoden auf den Inhalt
  - Vergleiche zwischen Strings
  - Teilzeichenfolgen
  - Suchen

Erzeugen eines Strings / Operatoren

Create:

- CString s1 = "Hallo ";
- CString s2 = s1
- CString s3("Hello“);
- CString _T(s4("Hello“)); // Unicode

Vergleiche:

- <
- >
- <=
- >=
- ==
- !=

Vergleiche:

- +=
- s1 = s2 + s3;
- compare
- compareNoCase
- int Collate *Vergleich, case sensitive*
- int CollateNoCase *Vergleich*

Methoden

- AnsiToOem
- OemToAnsi
- GetAt(i)
- SetAt(i)
- [] s1[2] = "A“;
- Left (int n);
- Right(int n);
- Mid(int first)
- Mid(int first, int nCount);
- Remove(TChar ch);
- Delete(int index, int nCount=1);

Methoden

- int GetLength() *Anzahl der Zeichen*
- Insert(int i, char) Insert(int i, string)
- Empty(); *Löschen des Inhalts, free buffer*
- bool IsEmpty();
- MakeLower()
- MakeUpper()
- TrimLeft()
- TrimRight()
- MakeReverse()
- GetBuffer Pointer auf den den Zero-String
- void Format(LPCTSTR lpszFormat, ...);
- void Format(UINT nFormatID, ...);

Methoden

- int Find(TCHAR ch) ; *Suchen eines Charakters*
- int Find(TCHAR ch, int nStart);
- int Find(LPCTSTR lpszSub); *Suchen eunes Strings*
- int Find(LPCTSTR pstr, int nStart);

Return Value
The zero-based index of the first character in this **CString** object that matches the requested substring or characters; -1 if the substring or character is not found.
Parameters

ch	A single character to search for.
lpszSub	A substring to search for.
nStart	The index to begin the search (0->n-1)
pstr	A pointer to a string to search for.

Beispiele

```
CString s1="ABCDEFGH";

s1.Left(3);             =>      ABC
s1.Right(3);            =>      FGH
s1.Mid(2);              =>      CDEFGH   // ab Zeichen 2
s1.Mid(3,2);            =>      DE
s1.Insert(3,"123");     =>      ABC123DEFGH

s1.Format("%s--%d", "ABC", 123);   =>      "ABC--123"
s2.Format("%s  %d", (LPCTSTR) s1, 123);
    setzt s2 auf "ABCDEFGH 123"

s.Find( 'C' );          =>      2
CString str("The stars are aligned");  // Suche ab Index
str.Find('e', 5);       =>      12
```

Klasse CFile

Eigenschaften:

- Lädt Daten aus Dateien
- Speichert Daten in Dateien
- Konstruktoren
  - Cfile cf(LPCTSTR lpszFileName, UINT nOpenFlags);
    oder
  - Cfile cf;
  - cf.open(lpszFileName, UINT nOpenFlags);
- Destruktoren:
  - close();

Klasse CFile: Flags

CFile::modeCreate	Neue Datei, Länge = Null
CFile::modeNoTruncate	Neue Datei, Länge beliebig
CFile::modeRead	Öffnen zum Lesen
CFile::modeReadWrite	Öffnen zum Lesen und Schreiben
CFile::modeWrite	Öffnen zum Schreiben
CFile::modeNoInherit	Kein Weitervererben möglich.
CFile::shareDenyNone	Alle Prozesse können zugreifen.
CFile::shareDenyRead	Kein anderer Prozess kann lesend zugreifen
CFile::shareDenyWrite	Kein anderer Prozess kann schreibend zugreifen.
CFile::shareExclusive	Kein anderer Prozess kann zugreifen.
CFile::typeText	Nur in CStdioFile und CMemFile.
CFile::typeBinary	Nur in CStdioFile und CMemFile.

Beispiel: Neue Datei schreiben

```
CFile cf;
if (cf.Open("C:\\1.Dat", CFile::modeWrite | CFile::modeCreate ) ) {
        char szData[]="ABCDEF";
        try {
          cf.Write(szData, strlen(szData) );
        }
        catch (CFileException *e) {
            // Error
        }
}
else {
    // Error
}
```


Beispiel: Neue Datei schreiben

```
CFile cf;
if (cf.Open("C:\\1.Dat", CFile::modeWrite | CFile::modeCreate ) ) {
        CString s = "ABCDEF";
        try {
          cf.Write(    (LPCTSTR) s,    s.GetLength() );
        }
        catch (CFileException *e) {
            // Error
        }}
else {
    // Error
}
```

Beispiel: Datei schreiben, ans Ende

```
CFile cf;
if (cf.Open("C:\\1.Dat", CFile::modeWrite ) ) {
        CString s = "ABCDEF";
        try {
          cf.SeekToEnd();
           cf.Write(    (LPCTSTR) s,    s.GetLength() );
        }
        catch (CFileException *e) {
            // Error
        }}
else {
    // Error
}
```


Beispiel:

```
CFile cf;
if (cf.Open("C:\\1.Dat", CFile::modeRead) ) {
    // lesen
    char cbBuffer[10000];
    try {
        int n = cf.Read(cbBuffer, sizeof(cbBuffer) );
    }
    catch (CFileException *e) {
        // Error
    }
}
else {
  // Error
}
```

Methoden der Klasse CFile

#include <afx.h>

Input/Output:

- Read — Liest Daten aus der Datei.
- Write — Schreibt Daten in eine Datei, ab der aktuellen Position.
- Flush — Alle Daten in internen Buffer werden in die Datei geschrieben.

Position:

- Seek(LONG lOff, UINT nFrom) — Position des Kursors setzen.
  - CFile::begin
  - CFile::current
  - CFile::end
- SeekToBegin — Position an den Anfang der Datei.
- SeekToEnd — Position an das Ende der Datei.
- GetLength — Ausgabe der Dateilänge in Bytes.
- SetLength — Ändern der Dateilänge.

Methoden der Klasse CFile

Status:

- DWord GetPosition() — Aktuelle Position.
- BOOL GetStatus(CFileStatus& rStatus);
- CString GetFileName() — Aktueller Dateinamen
- GetFileTitle — Aktueller Namen (*bsp1*.txt)
- GetFilePath — Komplette Adresse.
- SetFilePath — Setzen des Dateinamens (vorm Schreiben).

Statische Methoden:

- Rename(LPCTSTR lpszOldName, LPCTSTR lpszNewName)
- Remove(LPCTSTR lpszName)

Beispiel: Remove

```
//example for CFile::Remove
char* pFileName = "test.dat";
TRY  {
    CFile::Remove( pFileName );
}
CATCH( CFileException, e ){
    #ifdef _DEBUG
        afxDump << "File " << pFileName
           << " cannot be removed\n";
    #endif
}
END_CATCH
```


Beispiel: Schreiben Int, Double

```
CFile cf;
if (cf.Open("C:\\1.bin", CFile::modeWrite | CFile::modeCreate ) ) {
      CString s = "ABCDEF";
      int i1=23;
      int i2=-23;
      double d1=123.345;
      double d2=-123.345;
      try {
          cf.Write( &i1, sizeof(i1) );  // mit Pointer
          cf.Write( &d1, sizeof(d1) );
          cf.Write( &i2, sizeof(i2) );
          cf.Write( &d2, sizeof(d2) );
          cf.Close();
      }
      catch (CFileException *e) {
           // Error
      }
}
```

Beispiel: Schreiben Int, Double

```
CFile cf;
if (cf.Open("C:\\1.bin", CFile::modeRead ) ) {
        int i1=0;
        int i2=0;
        double d1=1;
        double d2=1;
        try {
         cf.Read( &i1, sizeof(i1) );
         cf.Read( &d1, sizeof(d1) );
         cf.Read( &i2, sizeof(i2) );
         cf.Read( &d2, sizeof(d2) );
           cf.Close();
        }
        catch (CFileException *e) {
          // Error
        }
}
```


Methoden der Klasse CStdioFile::CFile

```
// Einlesen einer Textes in einem Buffer
// Das Einlesen endet beim CR/LF
// Speicher muss vorhanden sein
```

- LPTSTR ReadString (LPTSTR lpsz, UINT nMax)
  Return Value:
  - Pointer eines Textes
  - NULL, wenn End-Of-File erreicht wurde

- BOOL ReadString(CString& rString);

- void WriteString(LPTSTR lpsz);

Methoden der Klasse CMemFile::CFile

- CMemFile erlaubt das Lesen und Schreiben in eine RAM-Datei.
- Die Funktionen sind „identisch“ der CFile-Methoden
- Einige Funktionen dürfen nicht benutzt werden
- Keinen Dateinamen
- Speicher wird automatisch verwaltet
- Speicher kann reserviert werden
- Mit dem Destruktor wird der Speicher freigegeben

Konstruktoren:

- CMemFile(UINT nGrowBytes = 1024);

- CMemFile(BYTE* lpBuffer, UINT nBufferSize,
  UINT nGrowBytes = 0);

Methoden der Klasse CMemFile::CFile

Beispiel:

```
CMemFile f;   // Instanz, sofort benutzbar, Speicher automatisch

BYTE * pBuf = (BYTE *)new char [1024];
CMemFile g( pBuf, 1024, 256 );
```

oder

```
BYTE * pBuf = (BYTE *)new char [1024];
CMemFile g;
g.Attach( pBuf, 1024, 256 );
```

CMemFile-Beispiel

Schreiben:

```
CMemFile cf;
int i1=23;
int i2=-23;
double d1=123.345;
double d2=-123.345;
cf.Write( &i1, sizeof(i1) );
cf.Write( &d1, sizeof(d1) );
cf.Write( &i2, sizeof(i2) );
cf.Write( &d2, sizeof(d2) );
```

Lesen:

```
i1=0;   i2=0;
d1=1;   d2=1;
cf.SeekToBegin();
try {
      cf.Read( &i1, sizeof(i1) );
      cf.Read( &d1, sizeof(d1) );
      cf.Read( &i2, sizeof(i2) );
      cf.Read( &d2, sizeof(d2) );
      cf.Close();
}
catch (CFileException *e) {
   // Error
}
```


Dynamische Datenspeicher

- **Felder**
- Listen
- HashMaps

Arrays bzw. Felder

- Klassisches C-Array, feste Größe
- MFC-Klasse CArray, dynamische Größe
- Eigenschaften:
  - Abfrage der Größe
  - Dynamisch, kein Datenverlust
  - Typsicher, Template

- Quicksort: http://support.microsoft.com/kb/216858/de

Methoden der Klasse CArray

Attribute:
GetSize();
Gets the number of elements in this array.
GetUpperBound()
Returns the largest valid index.
SetSize(**int** *nNewSize*, **int** *nGrowBy* = **-1**)
Sets the number of elements to be contained in this array.

Operations
FreeExtra();
Frees all unused memory above the current upper bound.
RemoveAll();
Removes all the elements from this array.

Methoden der Klasse CArray

Element Access
GetAt(**int** *nIndex*) [i]
Returns the value at a given index.

SetAt(**int** *nIndex*, *ARG_TYPE newElement*) [i]
Sets the value for a given index; array not allowed to grow.

ElementAt(**int** *nIndex*)
Returns a temporary reference to the element pointer within the array.

GetData(); // In Iterator-Schleife
Allows access to elements in the array. Can be NULL.

Methoden der Klasse CArray

Growing the Array
SetAtGrow(index, Value)
Sets the value for a given index; grows the array if necessary [i]
Add(value)
Adds an element to the end of the array; grows the array if necessary
Append(value)
Appends another array to the array; grows the array if necessary
Copy
Copies another array to the array; grows the array if necessary

Insertion/Removal
InsertAt(**int** *nIndex*, *ARG_TYPE newElement*, **int** *nCount* = **1)**
Inserts an element (or all the elements in another array) at a specified index.
RemoveAt(**int** *nIndex*, **int** *nCount* = **1)**
Removes an element at a specific index.

1. Beispiel CArray

- Methode FreeExtra
  - pArray.SetSize(50);
  - pArray.SetSize(30);
  - // 20 Elemente zuviel
  - pArray.FreeExtra(); // Freigabe

2. Beispiel CArray

- SDI-Grafik Labor
  - **Headerdatei**
  - CArray<CDblPoint, CDblPoint&> m_liste;
  - **CPP-Datei**:
  - m_liste.SetAtGrow(0, CDblPoint(2.2, 1.0));
  - m_liste.SetAtGrow(1, CDblPoint(3.2, 0.2));
  - void CLabor2_GrafikDoc::addPoint(double x, double y) {
  - CDblPoint pPoint;
  - int n=m_liste.GetCount();
  - m_liste.SetAtGrow(n, CDblPoint(x,y));
  - }

2. Beispiel CArray, Abfrage

```
CDblPoint CLabor2_GrafikDoc::GetPoint(int index) {
        CDblPoint point;
        int n=m_liste.GetCount();
        if ( (index>=0)  && (index<n) ){
                return m_liste[index];
        }
        else{
                return CDblPoint(-1,-1);
        }
}
```


2. Beispiel CArray, Serialize

```
n = m_liste.GetCount();
ar << n;
for (i=0; i<n; i++) {
  point = m_liste[i];
  ar << point.getX();
  ar << point.getY();
}
```

```
int x,y;
n = m_liste.GetCount();
ar >> n;
m_liste.RemoveAll();
for (i=0; i<n; i++) {
  ar >> x;
  ar >> y;
  m_liste.SetAtGrow( i, CDblPoint(x,y) );
}
```

Beispiel CArray

- Löschen
  - intArray.removeAt(0);
- Löschen
  - intArray.removeAt(0,3); // löschen 0,1,2
- Löschen
  - intArray.removeAll(); // ohneLöschen der Elemente

Beispiel CArray

- Löschen
  - // mit Löschen der Elemente
  - int n = pArray.GetSize();
  - for (int i=0; i<n; i++) {
  - delete pArray[i]; // löscht Speicher !!!!!
  - }
  - pArray.removeAll();

Beispiel vordefinierter Arraytypen

Arrays	**Lists**	**Maps**
CObArray	CObList	CMapPtrToWord
CByteArray	CPtrList	CMapPtrToPtr
CDWordArray	CStringList	CMapStringToOb
CPtrArray		CMapStringToPtr
CStringArray		CMapStringToString
CWordArray		CMapWordToOb
CUIntArray		CMapWordToPtr

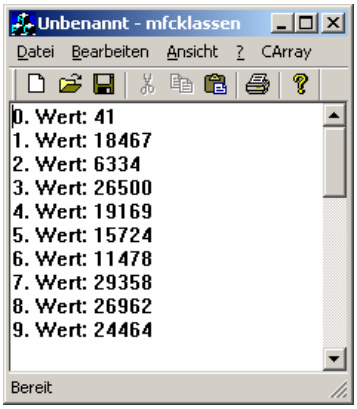
2. Beispiel CArray

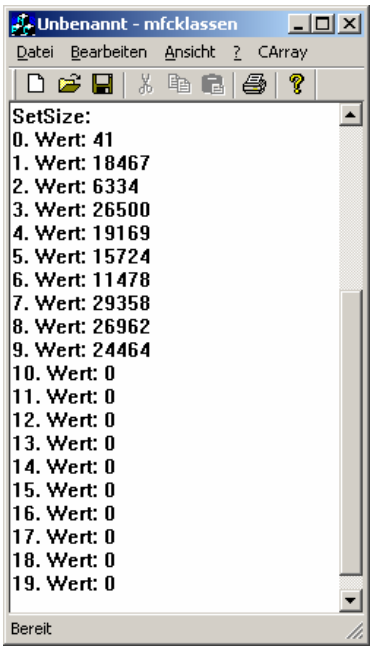
- Einfügen von zehn Elementen
- Vergrößern auf zwanzig Elementen
- Ausgabe

```
void CMfcklassenView::OnCarrayBsp2()
{
    int i;
    CUIntArray        intArray;        // vorgegeben
    intArray.SetSize(10);
    for (i=0; i<10;i++) {
        intArray[i] = rand();
    }
    CString sStr;
    CString sContent="";
    for (i=0; i<10;i++) {
        sStr.Format("%d. Wert: %d\r\n",i,intArray[i]);
        sContent += sStr;
    }
    CEdit * ed = &GetEditCtrl();
    ed->SetWindowText( sContent );
```



```
void CMfcklassenView::OnCarrayBsp2()
{
    // zweiter Teil
    intArray.SetSize(20);
    sContent += "\r\n\r\nSetSize:\r\n";
    for (i=0; i<20;i++) {
        sStr.Format("%d. Wert: %d\r\n",i,intArray[i]);
        sContent += sStr;
    }
    ed->SetWindowText( sContent );
}
```





Ergänzung

- Masken, Resourcen-String
- Felder
- **Listen**
- HashMaps

Listen

- Klassische C-Listen: einfach, doppelt verkettet
- MFC-Klasse CObList
- Eigenschaften:
  - Dynamisch
  - Kein Kopieren beim Vergrößern wie in Java
  - Typsicher, Template
  - CObList Cobject-Zeiger
  - CPtrList void-Zeiger
  - CStringList CString-Zeiger

Methoden der Klasse CObList

Head/Tail Access

CObject*& GetHead()
Returns the head element of the list (cannot be empty).

CObject*& GetTail();
Returns the tail element of the list (cannot be empty).

Methoden der Klasse CObList

Operations
RemoveHead();
Removes the element from the head of the list.
RemoveTail();
Removes the element from the tail of the list.

AddHead(**CObject*** *newElement*);
Adds an element (or all the elements in another list) to the head of the list (makes a new head).
AddTail(**CObject*** *newElement*);
Adds an element (or all the elements in another list) to the tail of the list (makes a new tail).
RemoveAll ();
Removes all the elements from this list.

Methoden der Klasse CObList

Iteration
POSITION GetHeadPosition();
Returns the position of the head element of the list.

POSITION GetTailPosition ();
Returns the position of the tail element of the list.

CObject*& GetNext(**POSITION&** *rPosition*)
Gets the next element for iterating.
Liefert den aktuellen Wert; rPosition wechselt zum nächsten !!
CObject*& GetPrev(**POSITION&** *rPosition*)
Gets the previous element for iterating.
Liefert den aktuellen Wert; rPosition wechselt zum vorherigen !!

Methoden der Klasse CObList

Retrieval/Modification
GetAt(**POSITION** *position*);
Gets the element at a given position.
SetAt(**POSITION** *pos*, **CObject*** *newElement*);
Sets the element at a given position.
RemoveAt(**POSITION** *position*);
Removes an element from this list as specified by position.

Insertion
InsertBefore(**POSITION** *position*, **CObject*** *newElement*);
Inserts a new element before a given position.
InsertAfter (**POSITION** *position*, **CObject*** *newElement*);
Inserts a new element after a given position.

Methoden der Klasse CObList

Searching

Find(**CObject*** *searchValue*, **POSITION** *startAfter* = **NULL**);
Gets the position of an element specified by string value.
FindIndex
Gets the position of an element specified by a zero-based index.

```
CObList list;
POSITION pos;     // CAge ist von CObject abgeleitet

list.AddHead( new CAge( 21 ) );
list.AddHead( new CAge( 40 ) );
      // List now contains (40, 21).
if( ( pos = list.FindIndex( 0 )) != NULL )
{
    ASSERT( *(CAge*) list.GetAt( pos ) == CAge( 40 ) );
}
```

Methoden der Klasse CObList

Status
GetCount();
Returns the number of elements in this list.
IsEmpty ();
Tests for the empty list condition (no elements).

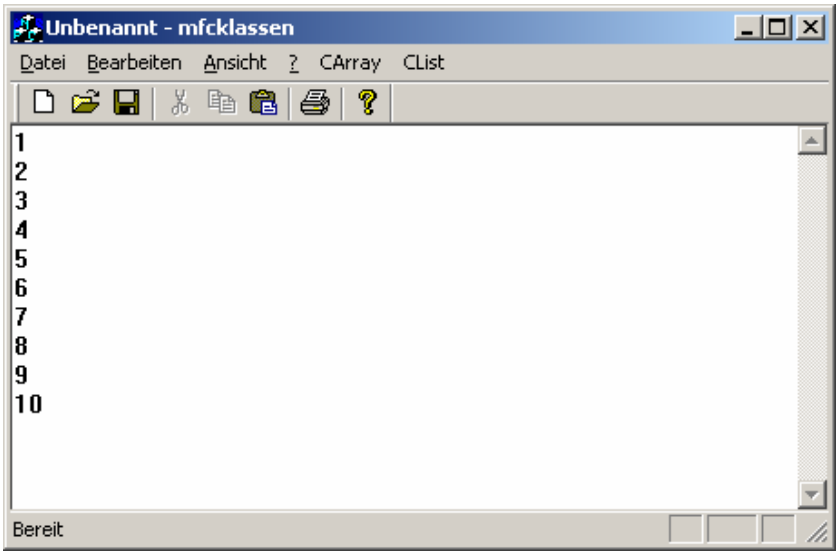
```
CObList list;

list.AddHead( new CAge( 21 ) );
list.AddHead( new CAge( 40 ) );
  // List now contains (40, 21).
ASSERT( list.GetCount() == 2 );
```


1. Listenbeispiel

- Erzeugen von 10. Strings (1 bis 10)
- Einfügen ans Ende
- Ausgabe der Elemente

```
void CMfcklassenView::OnClistBsp1() {
    CStringList myListe(20);  // default 10 Elemente
    CString * pStr;
    int i;
    for (i=0; i<10;i++) {
          pStr = new CString();
          pStr->Format("%d",i+1);
          myListe.AddTail( pStr->GetBuffer(100) );
    }
    CString sStr;
    CString sContent="";
    POSITION pos = myListe.GetHeadPosition();
    while (pos != NULL) {
          sStr = myListe.GetNext(pos);
          sContent += sStr + "\r\n";
    }
    CEdit * ed = &GetEditCtrl();
    ed->SetWindowText( sContent );
}
```

Beispiel für das Löschen eines Elementes

```
CObList list;
POSITION pos1, pos2;
CObject* pa;
list.AddHead( new CAge( 21 ) );
list.AddHead( new CAge( 40 ) );
list.AddHead( new CAge( 65 ) );
// List now contains (65 40, 21).
for( pos1 = list.GetHeadPosition();
   ( pos2 = pos1 ) != NULL; ) {
   if( *(CAge*) list.GetNext( pos1 ) == CAge( 40 ) ) {
      pa = list.GetAt( pos2 ); // Save the old pointer
      list.RemoveAt( pos2 );
      delete pa; // Deletion avoids memory leak.
   }
}
```


2. Listenbeispiel

- Erzeugen von 10. Strings (1 bis 10)
- Löschen des oberstes Elementes
- Löschen des vierten Elementes (0..4)
- Einfügen eines Elementes nach dem ersten Element

2. Listenbeispiel

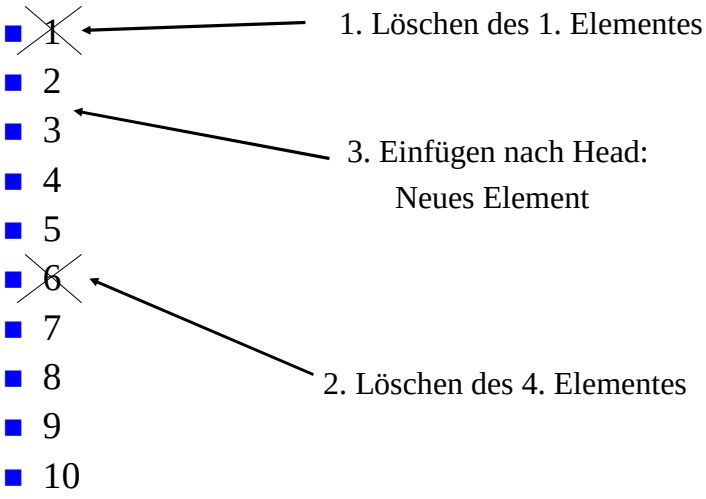
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

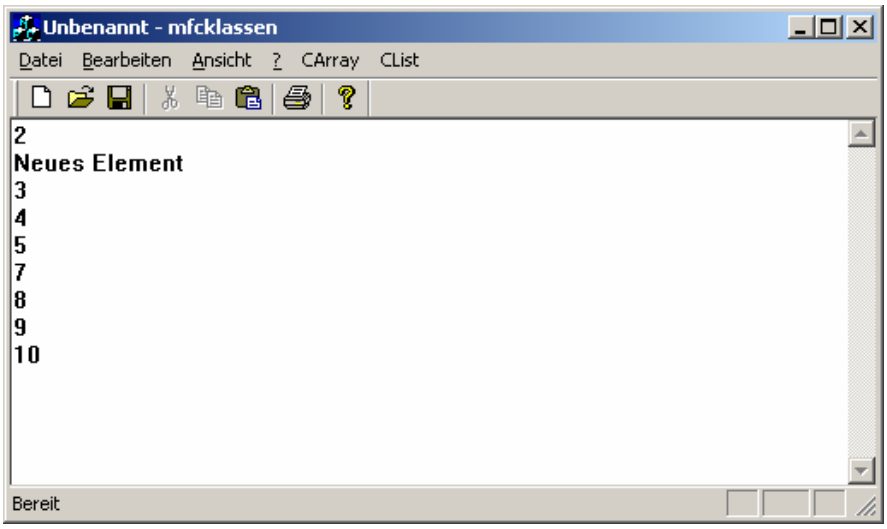
1. Löschen des 1. Elementes

3. Einfügen nach Head:
"Neues Element"

2. Löschen des 4. Elementes

2. Listenbeispiel






```
void CMfcklassenView::OnClistBsp2()
{
    CStringList myListe(20);  // default 10 Elemente
    CString * pStr;
    CString sStr;
    POSITION pos;
    int i;
    for (i=0; i<10;i++) {
          pStr = new CString();
          pStr->Format("%d",i+1);
          myListe.AddTail( pStr->GetBuffer(100) );
    }
    myListe.RemoveHead();
```

```
void CMfcklassenView::OnClistBsp2() {
    // 4. Element löschen, von 0 gezählt
    pos = myListe.GetHeadPosition();
    int count=-1;
        // GetNext gibt den aktuellen Wert zurück,
        // die Position geht aber zum nächsten Element
        // darum darf nur bis zum dritten Element gezählt werden
        // dann ist die 4. Position erreicht !!
    int posi = 4;
    while (pos != NULL) {
        count++;
        sStr = myListe.GetNext(pos);
        if (count == (posi-1) ) {
                 // nun ist pos auf den vierten Element
                sStr =  myListe.GetAt(pos);
                myListe.RemoveAt(pos);
                delete pStr;
                break;
        }
    }  // while
```



```
void CMfcklassenView::OnClistBsp2() {
        // Einfügen nach dem Head
    pStr = new CString("Neues Element");
    pos = myListe.GetHeadPosition();
    myListe.InsertAfter(pos, pStr->GetBuffer(100) );  // an 1. Position

        // Ausgabe
    CString sContent="";
    pos = myListe.GetHeadPosition();
    while (pos != NULL) {
        sStr = myListe.GetNext(pos);
        sContent += sStr+"\r\n";
    }
    CEdit * ed = &GetEditCtrl();
    ed->SetWindowText( sContent );
}
```

3. Listenbeispiel

- Erzeugen von 10. Punkten
- Einfügen in die Liste
- Ausgabe

- Erzeugen von 10. Punkten
- Einfügen in die Liste

```
void CMfcklassenView::OnClistBsp3() {
    CObList myListe;  // default 10 Elemente
    CPoint * pPoint;
    CString sStr;
    int i;
    for (i=0; i<10;i++) {
          pPoint = new CPoint( rand(), rand() );
          myListe.AddTail( (CObject*) pPoint );
    }
```

- Ausgabe

```
void CMfcklassenView::OnClistBsp3() {
    CObList myListe;  // default 10 Elemente
    CString sContent="";
    POSITION pos = myListe.GetHeadPosition();
    i=0; // Nummer für die Ausgabe
    while (pos != NULL) {
        pPoint = (CPoint *) myListe.GetNext(pos);
        sStr.Format("%d. x-Wert: %d   y-Wert: %d\r\n\r\n",
                i,pPoint->x, pPoint->y);
        sContent += sStr;
        i++;
    }
    CEdit * ed = &GetEditCtrl();
    ed->SetWindowText( sContent );
}
```


Ergänzung

- Masken, Resourcen-String
- Felder
- Listen
- **HashMaps**

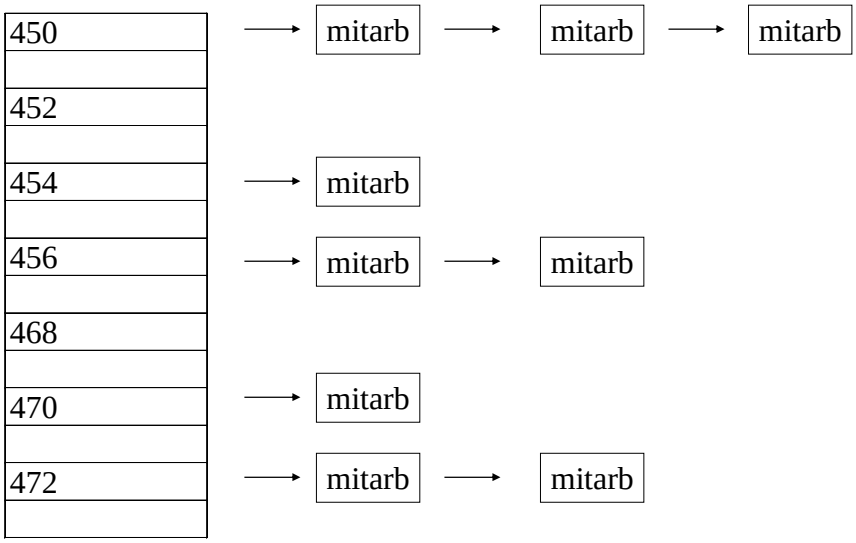
CMap, Hashmap

- Liste mit Id-Index
- MFC-Klasse CMap
- Eigenschaften:
  - Dynamisch
  - Schnelles Finden beim Suchen
  - Typsicher, Template
  - CMapWordToPtr Speichert void-Zeiger, key aus Word
  - CMapPtrtoWord Speichert Word, key aus void
  - CMapPtrToPtr Speichert void, key aus void
  - CMapWordToObj Speichert obj, key aus Word
  - CMapStringToObj Speichert obj, key aus String

Beispiel

- Klasse CMitarbeiter für eine Verwaltung
  - m_Name
  - m_PLZ
- Speicherung von ca. 100 Mitarbeitern
- HashMap von ca. 120 Einträge
- Definiere einen ID-Schlüssel
  - PLZ, eventuell doppelte Einträge !!
- Einfügen eines Mitarbeiters
  - hashmap.LookUp(mitarb.m_PLZ, mitarb);

Aufbau einer Hashmap



Methoden der Klasse CMap

Konstruktor
CMap(int *nBlockSize* **= 10);**

Operations
BOOL Lookup *ARG_KEY key***,** *VALUE***&** *rValue*);
Looks up the value mapped to a given key.

void SetAt (*ARG_KEY key***,** *ARG_VALUE newValue*);
Inserts an element into the map; replaces an existing element if a matching key is found.

*VALUE***&** operator [](*ARG_KEY key* **);** // identisch zu SetAt
Inserts an element into the map — operator substitution for SetAt.

Methoden der Klasse CMap

Operations
BOOL RemoveKey (*ARG_KEY key*);
Removes an element specified by a key.

void RemoveAll();
Removes all the elements from this map.

POSITION GetStartPosition ();
Returns the position of the first element.

void GetNextAssoc (**POSITION&** *rNextPosition*, *KEY***&** *rKey*, *VALUE***&** *rValue*);
Gets the next element for iterating.

Methoden der Klasse CMap

Operations
UINT GetHashTableSize();
Returns the size (number of elements) of the hash table.

void InitHashTable (**UINT** *hashSize*);
Initializes the hash table and specifies its size.

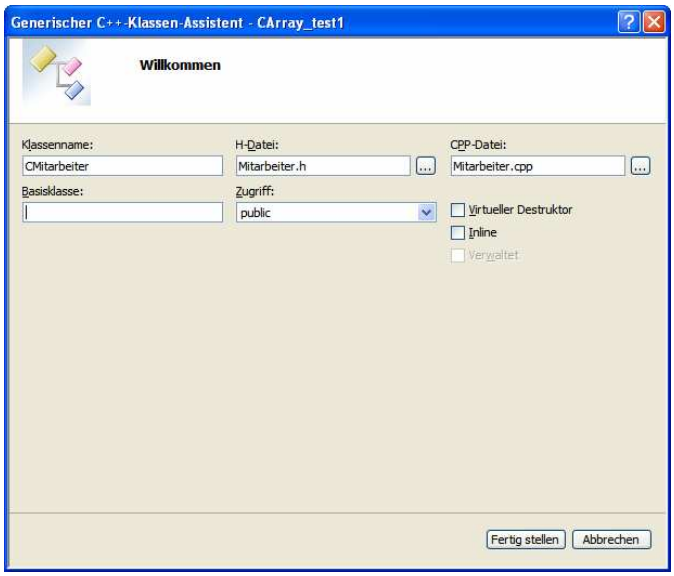
Status
int GetCount ();
Returns the number of elements in this map.

BOOL IsEmpty ();
Tests for the empty-map condition (no elements).

1. Hashmap-Beispiel

- Erzeugen von 10. Strings (1 bis 10) mit Nummern
- Einfügen ans Ende
- Ausgabe der Elemente
- Key muss eindeutig sein !

Neue Klasse



Headerdatei

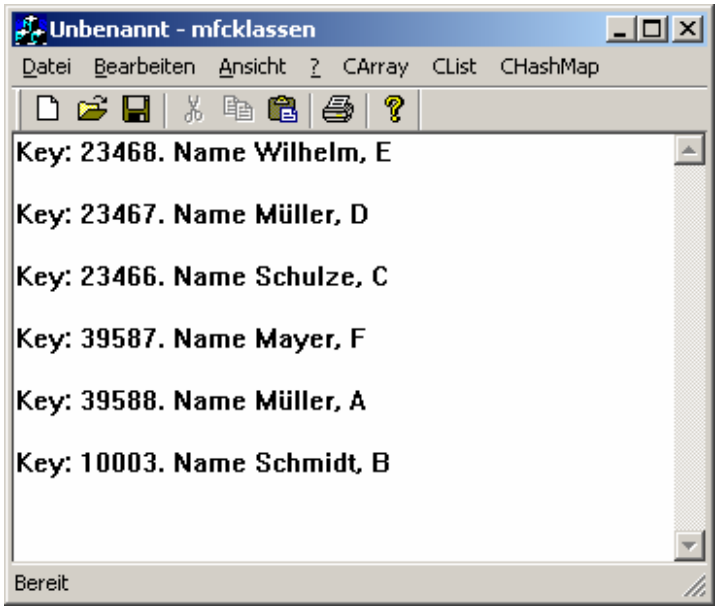
```
class CMitarbeiter
{
public:
        CString m_Name;
        UINT m_PLZ;
        CMitarbeiter();
        CMitarbeiter(CString Name, UINT PLZ);
        virtual ~CMitarbeiter();

};
```


CPP-Datei

```
CMitarbeiter::CMitarbeiter()
{
        m_Name = "";
        m_PLZ = 0;
}

CMitarbeiter::CMitarbeiter(CString Name, UINT PLZ) {
        m_Name = Name;
        m_PLZ = PLZ;
}
```




```
void CMfcklassenView::OnChashmapBsp1()  {
        CMapWordToOb myMap;
        CMitarbeiter * m;
        m = new CMitarbeiter("Müller, A",39588);
        myMap.SetAt( m->m_PLZ, (CObject *) m);

        m = new CMitarbeiter("Schmidt, B",10003);
        myMap.SetAt( m->m_PLZ, (CObject *) m);

        m = new CMitarbeiter("Schulze, C",23466);
        myMap.SetAt( m->m_PLZ, (CObject *) m);

        m = new CMitarbeiter("Müller, D",23467);
        myMap.SetAt( m->m_PLZ, (CObject *) m);
        m = new CMitarbeiter("Wilhelm, E",23468);
        myMap.SetAt( m->m_PLZ, (CObject *) m);

        m = new CMitarbeiter("Mayer, F",39587);
        myMap.SetAt( m->m_PLZ, (CObject *) m);
```

```
void CMfcklassenView::OnChashmapBsp1()  {
        // Ausgabe
    POSITION pos;
    unsigned short key;
    CString sStr;
    CObject * obj;
    CString sContent="";
    pos = myMap.GetStartPosition();
    while (pos != NULL) {
        myMap.GetNextAssoc(pos, key, obj);
        m = (CMitarbeiter *) obj;
        sStr.Format("Key: %d. Name %s\r\n\r\n",
                key, m->m_Name.GetBuffer(100) );
        sContent += sStr;
    }
    CEdit * ed = &GetEditCtrl();
    ed->SetWindowText( sContent );
}
```