EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

C++ für IT-Berufe

```
#include <iostream>
using namespace std;

int main()
{
     cout << "Informationsteil:" << endl;

     cout << " -  Strukturierte Programmierung
                  mit C++" << endl;

     cout << " -  Objektorientierte Programmierung
                  mit C++" << endl;

     cout << " -  Windows-Dektop-Programmierung
                  mit C++" << endl;

     cout << endl;
     cout << "Aufgabenpool" << endl;
     cout << endl;
     cout << "Lernsituationen" << endl;
     cout << endl;
     cout << endl;
     return 0;
}
```

3. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 42781 Haan-Gruiten

Europa-Nr.: 85498

Verfasser:
Dirk Hardy, 46049 Oberhausen

3. Auflage 2018

Druck 5 4 3 2 1

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-8545-0

http://www.europa-lehrmittel.de
Satz: Reemers Publishing Services GmbH, Krefeld
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagfotos: matttilda-fotolia.com, Gina Sanders-fotolia.com
Druck: Medienhaus Plump, 53619 Rheinbreitbach

Vorwort

Die Entwicklung von Anwendungen geht heutzutage weit über das reine Programmieren hinaus. Die Anforderungen an den Fachinformatiker bzw. Informatiker mit der Fachrichtung Anwendungsentwicklung sind sehr komplex geworden. Es reicht nicht mehr aus, mit einer speziellen Programmiersprache zu arbeiten. Vielmehr sind umfangreiche Kenntnisse in mehreren Programmiersprachen, in Datenbankmanagementsystemen und weiteren Gebieten wie der Client-Server-Programmierung, Web-Programmierung oder auch App-Programmierung nötig. Das Gemeinsame dieser verschiedenen Aspekte ist die Objektorientierung. Die modernen Programmiersprachen sind objektorientiert, die Datenbankmanagementsysteme werden in Zukunft verstärkt objektorientiert sein. Mischformen wie objektrelationale Datenbanken sind schon länger im Einsatz. Eine gute Grundlage für diese Anforderungen ist die Erlernung der Sprache C++. Die Sprache ist plattformunabhängig und objektorientiert. Sie ist schwerer zu erlernen als andere Sprachen, bietet allerdings dafür einige Vorteile:

- Geschwindigkeit
- Weite Verbreitung
- Sehr viele Tools und Bibliotheken
- Compiler und Entwicklungswerkzeuge auf verschiedensten Plattformen
- Grundlage vieler weiterer Sprachen wie Java, Javascript, Perl oder auch PHP und C#

Die letzte ISO-Standardisierung der Sprache C++ war 2017. Damit verfügt C++ über viele neue Konzepte, die in anderen modernen Programmiersprachen (wie C#) bereits umgesetzt worden sind. Die vorliegende dritte Auflage dieses Buches nimmt diese Neuerungen in einem eigenen Kapitel auf und geht dann auch intensiver auf die STL (Standard Template Library) ein, die mit den Standards C++11, C++14 und C++17 weiter ausgebaut wurde.

Den Lesern dieses Buches bleibt viel Erfolg bei der Erlernung von C++ zu wünschen und mit einem Zitat des Erfinders von C++ (Bjarne Stroustrup) zu schließen: „Die Frage *Wie schreibt man ein gutes C++ -Programm?* hat sehr viel Ähnlichkeit mit der Frage *Wie schreibt man gute englische Prosa?* Darauf gibt es zwei Antworten: Wisse, was Du sagen willst und übe. Halte Dich an gute Vorbilder. Beide Antworten gelten für C++ ebenso wie für Englisch – und sind ebenso schwer zu befolgen."

Für Anregungen und Kritik zu diesem Buch sind wir Ihnen dankbar (gerne auch per E-Mail).

Dirk Hardy
E-Mail: Hardy@DirkHardy.de

Im Oktober 2018

Verlag Europa-Lehrmittel
E-Mail: Info@Europa-Lehrmittel.de

Aufbau des Buches

Das vorliegende Buch möchte die Sprache C++ möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **fünf Teile** getrennt. Die ersten drei Teile des Buches dienen als **Informationsteil** und bieten eine systematische Einführung in die **strukturierte Programmierung als auch in die objektorientierte Programmierung mit C++.** Abgerundet wird diese Einführung mit einem **Einstieg in die klassische Windows-Programmierung,** die eine hervorragende Grundlage für alle GUI-Programmierungen bietet.

Der vierte Teil des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen.

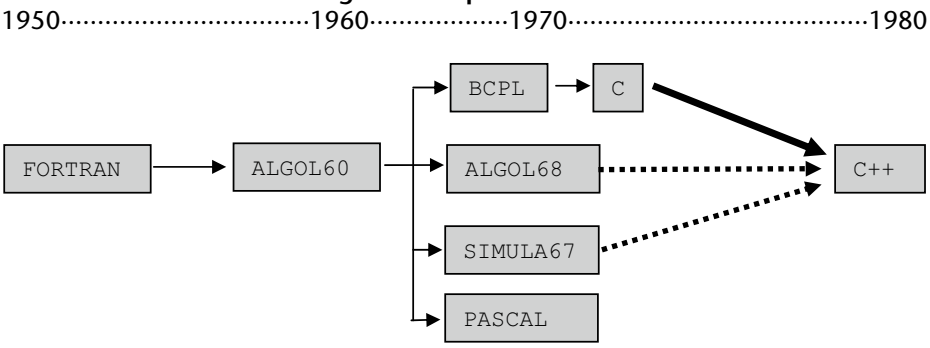
Der **fünfte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Lernfeld Entwickeln und Bereitstellen von Anwendungssystemen aus dem Rahmenlehrplan für die IT-Berufe (speziell Fachinformatiker-Anwendungsentwicklung). Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen (speziell Fachinformatiker), aber auch von den informationstechnischen Assistenten genutzt werden.

Als Entwicklungswerkzeug wird in diesem Buch die ***Community- Edition* Visual Studio 2017** von Microsoft genutzt. Diese Entwicklungsumgebung ist kostenfrei als Download im Internet verfügbar.

Dirk Hardy Oktober 2018

1.1.3 Kleiner Stammbaum der Programmiersprachen



C++ im Überblick:

- Entwickelt von Bjarne Stroustrup in den 80er-Jahren.
- Basiert auf der Sprache C und hat Konzepte der Sprachen SIMULA67 und ALGOL68 übernommen.
- C++ ist eine objektorientierte Sprache.
- Sie ist plattformunabhängig, systemnah und sehr schnell.
- Viele Betriebssysteme wurden in C++ programmiert.
- Die Sprache ist sehr weit verbreitet und kommt in technischen, kaufmännischen und wissenschaftlichen Anwendungen zum Einsatz.

1.2 Bestandteile eines C++-Programms

1.2.1 Was ist ein Programm?

Ein Programm besteht aus einer Folge von Anweisungen an den Computer. Diese Anweisungen (Befehle) werden in einer bestimmten Form gegeben – und zwar in einer speziellen Sprache (Programmiersprache).

Diese Programmiersprache wird vom Computer nicht direkt verstanden, sondern muss erst „übersetzt" werden, und zwar in eine für den Computer verständliche Sprache, den **Maschinencode.**

Nach der Übersetzung kann das Programm gestartet werden.

```
Anweisung 1;
Anweisung 2;
Anweisung 3;
Anweisung 4;
Anweisung 5;
:
:
:
Anweisung N;
```

Eine endliche Folge von eindeutigen Anweisungen an den Computer nennt man **Algorithmus.**

1.2.2 C++-Quellcode

Die Anweisungen an den Computer in der Sprache C++ werden in Dateien gespeichert. Der Inhalt dieser Dateien wird als **Quellcode** bezeichnet.

Normalerweise haben C++-Quellcode-Dateien die Endungen

- `cpp` für Cplusplus
- `h bzw. hpp` für Header bzw. Header-plusplus.

2 Das erste C++-Programm

2.1 Ausgabe auf dem Bildschirm

Für die Ausgabe auf dem Bildschirm sind einige Vorbereitungen zu treffen.

Neben dem Kennenlernen der Entwicklungsumgebung, damit überhaupt ein Programm geschrieben werden kann, müssen noch weitere Punkte geklärt werden:

- Wie werden ein C++-Projekt und eine Quellcode-Datei angelegt?
- Welche Bibliothek muss für die Ausgabe auf dem Bildschirm eingebunden sein?
- Wie sieht das „Hauptprogramm“ aus?
- Mit welcher Anweisung wird eine Bildschirmausgabe erreicht?

Die folgenden Kapitel geben Aufschluss über diese Punkte.

2.1.1 Ein C++-Projekt in Visual Studio anlegen

Die integrierte Entwicklungsumgebung ***Visual Studio 2017*** ist eine komfortable Umgebung, um C++-Programme zu entwickeln. Besonders erfreulich ist der Umstand, dass die Umgebung kostenfrei als *Community-Edition* im Internet bereit steht. Ein C++-Programm besteht aus einer oder mehreren Quellcodedateien. Diese Dateien werden in einem Projekt organisiert.

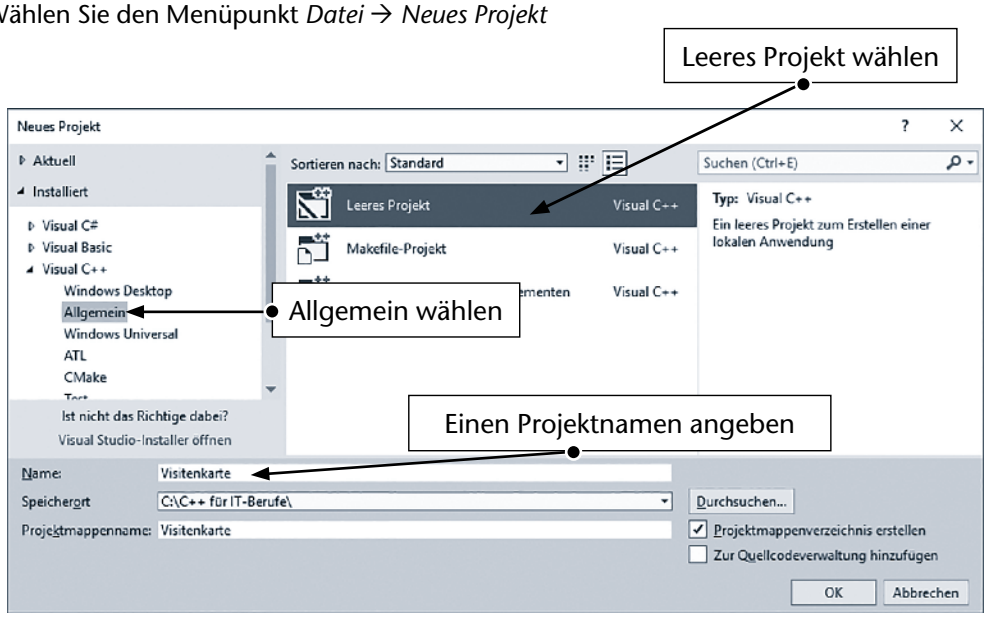
Visual C++ unterscheidet im Prinzip folgende Projektarten:

- Windows-Desktopanwendungen (eine Windows-Anwendung)
- **Windows-Konsolenanwendung / leere Anwendung (ähnlich einem DOS-Programm)**
- Universal Windows Platform – Anwendungen (Apps)
- Dynamic Link Library (Bibliotheksanwendung)

In diesem Buch ist hauptsächlich eine Projektform von Bedeutung: **die Windows-Konsolenanwendung bzw. die leere Anwendung**. Diese Form ist ausreichend, um die Programme in C++ auf dem Bildschirm darzustellen. Die Konsolenanwendung ist natürlich nicht so ansprechend wie ein Windows-Programm oder eine App, aber, um die Sprache C++ zu lernen, völlig ausreichend. Zum Abschluss des Buches wird die Windows-Desktopanwendung eingeführt, mit der klassische Windows-Programme realisiert werden können.

Anlegen eines neuen Projektes:

- Starten Sie ***Visual Studio 2017***
- Wählen Sie den Menüpunkt *Datei → Neues Projekt*



5 Selektion und Iteration

Die Selektion (Auswahl) und die Iteration (Wiederholung) sind zwei sehr wichtige Konstrukte in einer Programmiersprache. In den vorherigen Kapiteln sind schon einige Probleme ohne diese Konstrukte gelöst worden, aber große und immer komplexer werdende Programme können ohne Selektion und Iteration nicht auskommen.

5.1 Die Selektion

5.1.1 Darstellung der Selektion mit einem Programmablaufplan

Problemstellung:

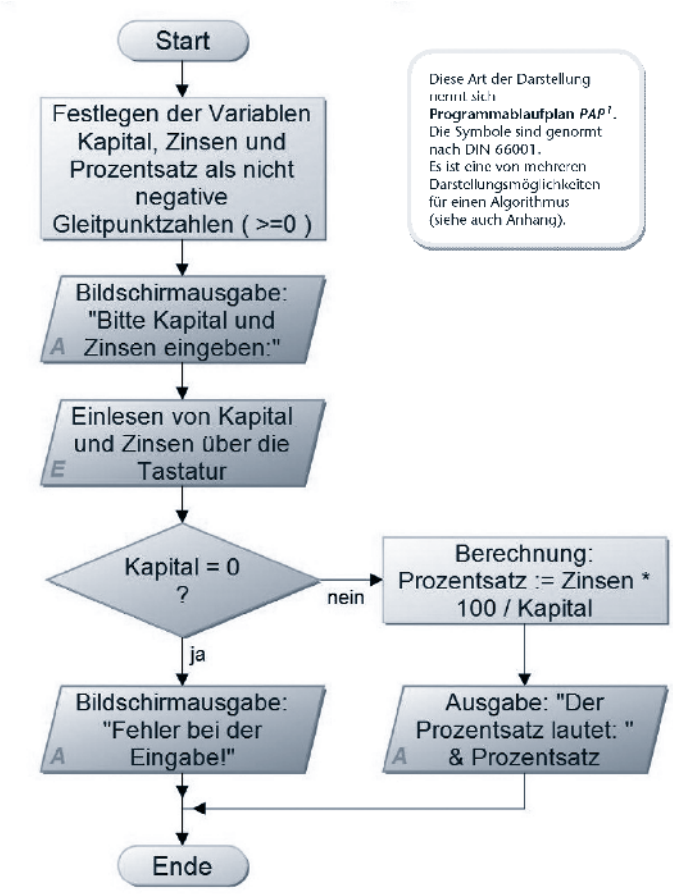
Es soll eine Berechnung des Prozentsatzes bei gegebenem Kapital und Zinsen durchgeführt werden.

Formel: $p = \frac{Z * 100}{K}$

Das Programm kann einen Fehler verursachen, wenn für das Kapital der Wert Null eingegeben wird (die Division durch Null ist verboten).

Lösungsmöglichkeit:

Das Programm erkennt, ob eine Null eingegeben wurde, und führt dann keine Berechnung durch. Das Problem wird durch eine Dokumentationstechnik, den Programmablaufplan, zuerst schematisch erfasst. Anschließend wird dann auf die Umsetzung in C++ eingegangen.



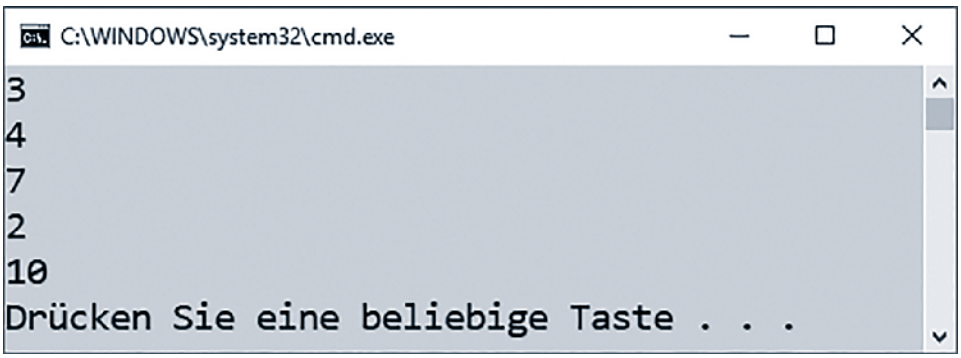
1 Der Programmablaufplan wurde mit dem Programm *PapDesigner* erstellt. Diese Software wurde speziell für die Ausbildung im IT-Bereich entwickelt und steht kostenfrei im Internet zum Download bereit.

Beispiel:

```
#include <iostream>
using namespace std;

int main()
{
   int Werte[5] = { 3, 4, 7, 2, 10 };

   for ( auto wert : Werte)
   {
          cout << wert<< endl;
   }

   return 0;
}
```

Das Array wird vom ersten bis zum letzten Element durchlaufen- Die Variable `wert` erhält automatisch den korrekten Datentyp. Auf die Variable ist allerdings nur lesend zugreifbar.

Bildschirmausgabe des Programmlaufs:



7.1.2 Mehrdimensionale Arrays

Die Vorstellung von mehrdimensionalen Arrays ist gerade am Anfang relativ schwer. Deshalb ist es sinnvoll, zuerst eine Vorstellung von einem zweidimensionalen bzw. dreidimensionalen Array zu bekommen.

Die Vorstellung eines zweidimensionalen Arrays entspricht einer Tabelle. Tabellen sind allgemein bekannt: Sie bestehen aus Zeilen und Spalten, die den entsprechenden Index des Arrays widerspiegeln.

Beispiel:

Es wird ein zweidimensionales Array angelegt.

```
float  Tab  [ 3 ] [ 4 ];
```

entspricht der Anzahl der Spalten

entspricht der Anzahl der Zeilen

	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	77.5
Zeile 1	124	99.99	453	67.89
Zeile 2	12	90.2	2727.5	22

```
Tab  [ 1 ] [ 2 ] == 453
```

Spaltenangabe

Zeilenangabe

Dreidimensionale Arrays kann man sich als eine Sammlung von Tabellenblättern vorstellen, die hintereinander angeordnet sind.

```
                        cin >> Knoten->Rechts->Inhalt;
                        Knoten->Rechts->Links = NULL;
                        Knoten->Rechts->Rechts = NULL;
                    }
                    break;
}
}while (wahl != 3);
```

4. Die Funktion `menu`:

Je nachdem, ob es bereits einen Eintrag bei Vater bzw. Mutter gibt, wird die Bildschirmausgabe gestaltet.

```
int menu(TKnoten * Aktuell)
{
      cout << "\t\t" << Aktuell->Inhalt;
      cout << endl;
      if (Aktuell->Links != NULL)
           cout <<"\t" << Aktuell->Links->Inhalt;
      else cout << "\t Vater?";
      if (Aktuell->Rechts != NULL)
           cout <<"\t\t" << Aktuell->Rechts->Inhalt;
      else cout << "\t\t Mutter?";
      cout << endl;
      cout << "<0> Zur Wurzel gehen" << endl;
      if (Aktuell->Links != NULL)
      cout <<"<1> Weitergehen zum Vater"    << endl;
      else cout <<"<1> Eingabe des Vaters" << endl;
      if (Aktuell->Rechts != NULL)
           cout <<"<2> Weitergehen zur Mutter" << endl;
      else cout <<"<2> Eingabe der Mutter" << endl;
      cout << "<3> Ende" << endl;
      cout << "Ihre Eingabe bitte: " << endl;
      int wahl;
      cin >> wahl;
      return wahl;
}
```

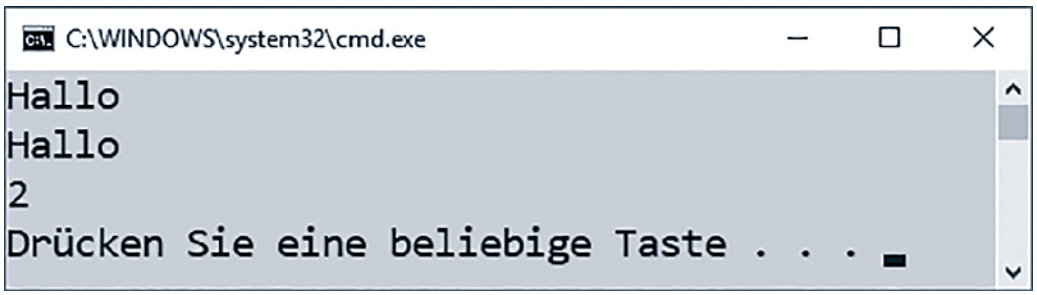
Hinweis:

Die höheren Datenstrukturen wie Liste und Baum haben Vorteile bezüglich der beliebigen Speicherung von Daten. Das Einfügen und Löschen ist bei der Liste besonders einfach. Das Suchen hingegen aufwändig, da die Liste immer sequenziell durchlaufen werden muss. Bei einem Baum erfolgt die Suche nach einem Element schneller, allerdings muss der Baum dann auch weiteren Bedingungen genügen – das führt schnell zu komplexeren Theorien wie dem AVL-Baum, auf die hier nicht weiter eingegangen wird.

Bevor man in einem Programm eine Liste oder einen Baum implementiert, sollte man prüfen, ob nicht ein gewöhnliches Array mit dynamischer Speicherverwaltung ausreicht, um die Daten zu speichern. Der Zugriff ist jedenfalls deutlich schneller und die Handhabung über den Indexoperator weniger mühsam.

Mithilfe des Indexoperators und der Methode `substr()` wird der String in den beiden `for`-Schleifen zeichenweise ausgegeben.

Die Methode `c_str()` liefert einen Zeiger vom Typ `char`. Damit können dann alle C-Zeichenkettenfunktionen benutzt werden.



15.3.2 Intelligente Zeiger

Die dynamische Speicherreservierung liegt in den Händen des Entwicklers, da C++ kein verwalteter Code ist und deshalb auch kein *garbage-collector* wie in Java oder C# für die Aufräumarbeiten sorgt. Dementsprechend ist die Speicherreservierung immer eine Quelle von Fehlern. Die intelligenten Zeiger sorgen dafür, dass dieses Problem entschärft wird. Dazu werden neue Arten von Zeigern zu Verfügung gestellt: `unique_ptr` und `shared_ptr`.

Der unique_ptr:

Der `unique_ptr` sorgt dafür, dass ein Zeiger immer nur auf ein dynamisches Objekt verweisen kann. Damit ist ausgeschlossen, dass es zusätzliche Verweise auf ein Objekt im Speicher gibt. Das folgende Beispiel zeigt die Verwendung eines solchen Zeigers:

Beispiel:

```
#include <memory>
#include <iostream>

using namespace std;

void Test()
{


   unique_ptr<double> zeiger (new double(1.5));
}

int main()
{
   unique_ptr<int> zeiger_1 (new int(10));
   unique_ptr<int> zeiger_2;


   //zeiger_2 = zeiger_1;


   if (zeiger_1) cout << "Zeiger_1 ist gueltig!" << endl;
   else cout << "Zeiger_1 ist nicht gueltig!" << endl;

   cout << "Inhalt von Zeiger_1: " << *zeiger_1 << endl;


   zeiger_1.reset();
```

Einbinden der Header-Datei **<memory>**!

Ein `unique_ptr` erhält ein dynamisch erstelltes Objekt vom Typ `double`. **Wenn die Funktion beendet wird, wird auch der lokale `unique_ptr` gelöscht und damit automatisch auch das `double`-Objekt.**

Zwei `unique_ptr` werden angelegt. Der erste Zeiger erhält ein dynamisch erstelltes Objekt vom Typ `int`.

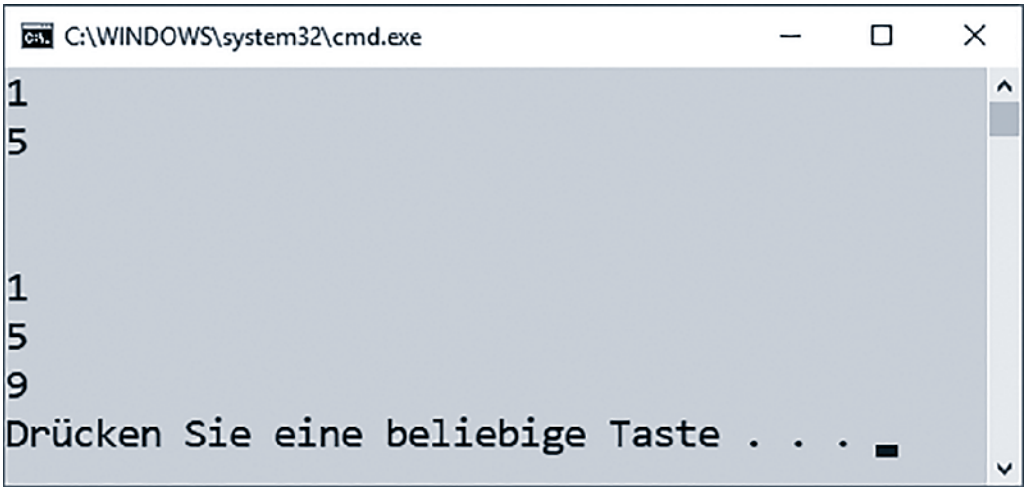
FEHLER: Der Compiler verhindert die Zuweisung eines `unique_ptr` auf einen anderen `unique_ptr`.

Wie gewohnt mit dem Inhaltsoperator arbeiten.

Den Speicher explizit freigeben!

```
        for (int i = 0; i < intContainer.capacity(); i++)
                                    cout << intContainer[i] << endl;
        return 0;
}
```

Nach dem Starten erscheint die folgende Ausgabe:



Der `vector` speichert zuerst zwei Werte, danach wird der Platz erhöht und ein weiterer Wert wird zugewiesen.

Das folgende Beispiel zeigt die Verwendung der `map`-Klasse:

Beispiel:

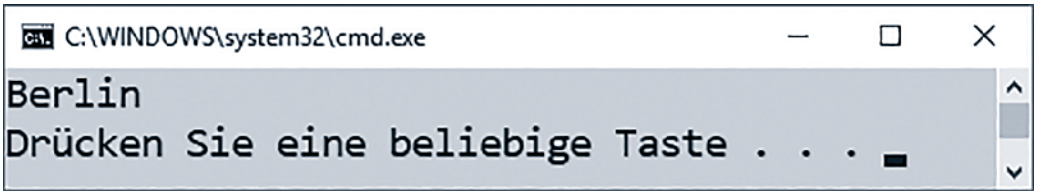
Header einbinden

```
#include <map>
#include <iostream>
#include <string>

using namespace std;

int main()
{
  map < string, string > testMap;

  testMap["Deutschland"] = "Berlin";
  testMap["Italien"] = "Rom";
  testMap["Frankreich"] = "Paris";

  cout << testMap["Deutschland"] << endl;

  return 0;
}
```

Es wird eine Instanz der Template-Klasse `map` gebildet. Schlüssel und Element sind beide vom Typ `string`.

Einfügen von Elementen mit entsprechenden Schlüsseln.

Nach dem Starten erscheint die folgende Ausgabe:



Das Beispiel zeigt die Verwendung eines assoziativen Arrays. Die Hauptstädte sind die `string`-Elemente, die über einen Schlüssel (Land) abrufbar sind.

```
   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}
```

Ansonsten bringt **ShowWindow** das Fenster auf den Bildschirm und **UpdateWindow** sorgt für die erste Aktualisierung.

Die Fensterprozedur – das Herz der Anwendung: hier werden alle Ereignisse (Benutzeraktionen, Systemnachrichten) verarbeitet und der Inhalt des Fensters entsprechend dargestellt.

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
                         wParam, LPARAM lParam)
{
    switch (message)
    {
```

Der Kern der Funktion ist eine große `switch`-Anweisung, die auf die Nachrichten in den entsprechenden `case`-Blöcken reagiert.

Die Variable **message** enthält Nachrichten vom Typ **WM**... (= Windows Message ...). Die **WM_COMMAND**-Nachricht zeigt eine Benutzereingabe an (z.B. Auswahl eines Menüpunktes).

```
      case WM_COMMAND:
      {
         int wmId = LOWORD(wParam);

         switch (wmId)
         {
```

Der Benutzer hat die Infobox aus dem Menü gewählt. Mit der Funktion **DialogBox** wird sie angezeigt.

```
               case IDM_ABOUT:
                     DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd,
                     About);
               break;
```

Der Benutzer hat die „Beenden“ aus dem Menü gewählt. Mithilfe der Funktion DestroyWindow wird die Nachricht **WM_DESTROY** an das Betriebssystem gesendet. Das Betriebssystem reiht die Nachricht in die Nachrichtenschlange ein.

```
               case IDM_EXIT:
                     DestroyWindow(hWnd);
               break;
```

Die Funktion **About** zeigt die „Infobox“ an. So wie in der Fensterprozedur werden auch hier Nachrichten ausgewertet.

```
INT_PTR CALLBACK About   (HWND hDlg, UINT message, WPARAM
                          wParam,LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
       case WM_INITDIALOG:
        return (INT_PTR)TRUE;

       case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}
```

16.2 Nachrichtenbearbeitung

16.2.1 Die Nachricht WM_PAINT

Die Nachricht WM_PAINT wurde bereits in der Darstellung des ersten Programms erläutert. Diese Nachricht ist aber so wichtig, dass noch einmal speziell darauf eingegangen werden muss: Wenn ein Fenster in irgendeiner Form verdeckt bzw. überdeckt wird, so wird genau dann die Nachricht WM_PAINT gesendet, wenn das Fenster erneut den Focus erhält. Ebenso wird die WM_PAINT gesendet, wenn das Fenster in seiner Größe geändert wird. **Dadurch wird sichergestellt, dass der Inhalt eines Fensters immer auf dem aktuellen Stand gehalten wird.** Im Gegensatz zu Konsolen-Applikationen ist es also sehr wichtig, dass das Programm jederzeit den gültigen Fensterinhalt ausgeben kann – und zwar angepasst an mögliche Veränderungen wie eine Größenänderung.

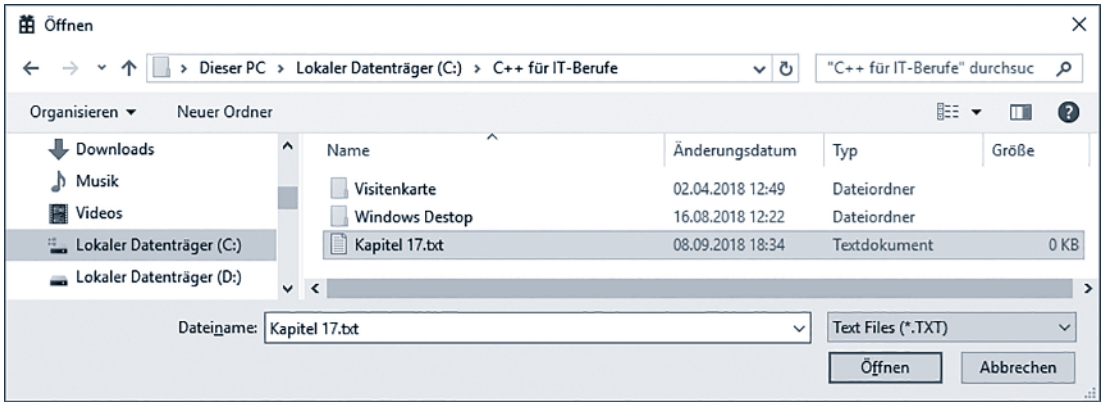
Das folgende Beispiel zeigt, wie oft diese Nachricht ausgesendet wird, indem jeder Aufruf der Nachricht gezählt wird und auf dem Bildschirm angezeigt wird. Dazu kann eine API-Funktion verwendet werden, die eine Art Meldungsfenster (MessageBox) anzeigt:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                         WPARAM wParam, LPARAM lParam)
{
       static int counter = 1;
:
:
:
:
case WM_PAINT:
{
       PAINTSTRUCT ps;
       HDC hdc = BeginPaint(hWnd, &ps);

       WCHAR Text[100+1];
```

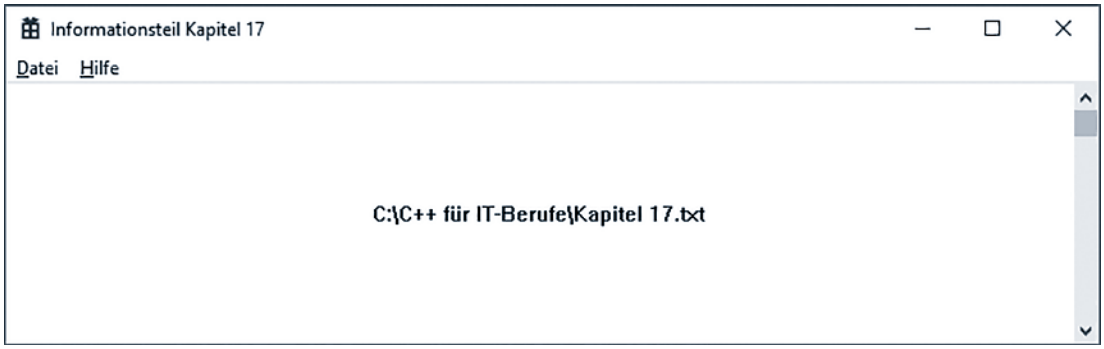
Statt einer globalen Variablen für einen Zähler, kann eine lokale Variable als ***static*** deklariert werden. Damit ist die Sichtbarkeit lokal, aber die Variable behält ihren Wert über die Funktionsaufrufe hinweg.

Ein statisches Zeichenketten-Array vom Typ WCHAR (Unicode).

Nach dem Starten öffnet sich sofort der Datei-Öffnen-Dialog:



Nach der Auswahl einer Textdatei wird der Name (incl. Pfad) im Clientbereich angezeigt:



17.2.2 Schriftwahl-Dialog

Mit dem `ChooseFont`-Dialog kann eine der installierten Schriftarten unter Windows ausgewählt werden. Dazu muss eine Struktur vom Typ `CHOOSEFONT` mit Werten gefüllt werden und an den Dialog übergeben werden. Der Dialog füllt anschließend diese Strukturvariable mit den Werten der gewählten Schriftart. Die API-Funktion `ChooseFont` ist wir folgt deklariert:

```
BOOL ChooseFont(LPCHOOSEFONT lpcf);
```

Die Funktion liefert ein `TRUE` zurück, wenn der Benutzer eine Schriftart gewählt und den OK-Button geklickt hat.

Das Füllen der `CHOOSEFONT`-Struktur ist etwas aufwändiger. Deshalb ist es sinnvoll die Struktur in einer eigenen Header-Datei etwas zu modifizieren. In der `CHOOSEFONT`-Struktur muss auch eine `LOGFONT`-Struktur angegeben werden, die ebenfalls durch den Aufruf der Funktion gefüllt wird. Anschließend kann diese Struktur-Variable dann für das Einsetzen der eigentlichen Schrift in den Geräte-Kontext genutzt werden.

Die folgende Header-Datei (FontDlg.h) zeigt diese Modifizierung:

```
#include <commdlg.h>
#include "stdafx.h"
CHOOSEFONT cf;
LOGFONT lf;

void DialogInit(HWND hwnd)
{
     cf.lStructSize  = sizeof (CHOOSEFONT);
     cf.hwndOwner    = hwnd;
     cf.hDC          = NULL;
     cf.lpLogFont    = &lf;
     cf.iPointSize   = 0;
     cf.Flags        = CF_INITTOLOGFONTSTRUCT |
                       CF_SCREENFONTS | CF_EFFECTS;
     cf.rgbColors    = 0;
     cf.lCustData    = 0;
     cf.lpfnHook     = NULL;
```

Header-Datei für die Dialoge einbinden.

Globale Variablen (der Einfachheit halber) für die Schriftart-Daten.

Strukturvariable für die Schriftart angeben!

```
Ihre Wahl: 3

Verkauf eines Power-PCs:
Bitte geben Sie die Modembezeichnung ein: Fritz 1.5
Bitte geben Sie die Grafikkarte ein: Hercules 6
Bitte geben Sie den Speicher ein: 512 MB
Bitte geben Sie den Prozessor ein: Intel 2.2 GHz
Bitte geben Sie die Kundenummer ein: 4443

Ihre Wahl: 4

Verkauf eines Server-PCs:
Wie viele Netzwerkkarten?: 2
Bitte geben Sie die 1. Karte ein: MAXCOM 1
Bitte geben Sie die 2. Karte ein: MAXCOM 2
Bitte geben Sie den Speicher ein: 1024 MB
Bitte geben Sie den Prozessor ein: Intel 3.2 GHz
Bitte geben Sie die Kundenummer ein: 3337

Ihre Wahl: 5

Liste der verkauften PC-Systeme:
1. Multimedia-PC an Kunde: 1737
2. Internet-PC an Kunde: 3428
3. Power-PC an Kunde: 4443
4. Server-PC an Kunde: 3337
```

Aufgabe 14.2

Finden Sie drei sinnvolle verschiedene Beispiele für Vererbungshierarchien, die eine abstrakte Basisklasse haben.

15 Aufgaben zur fortgeschrittenen Programmierung

Aufgabe 15.1

Entwicklen Sie eine Klasse `CList`, die ein Container für verschiedene Datentypen sein soll. Nutzen Sie dazu die Klassen-Templates unter C++.

Beispiel eines möglichen Hauptprogrammes:

```
int main()
{
      CList <float> FloatListe;
      FloatListe.Einfuegen(10.5);
      FloatListe.Einfuegen(22.44);
      FloatListe.Einfuegen(12.12);

      for ( int i = 0 ; i < FloatListe.Anzahl() ; i++)
           cout << FloatListe[i];

      return 0;
}
```

Hinweise:

- Das Speichern der Elemente soll in Form einer verketteten Liste erfolgen.
- Benutzen Sie die Ausnahmebehandlung, um Speicherfehler und andere Fehler professionell abzufangen.
- Überladen Sie den Indexoperator, so dass ein Zugriff wie bei gewöhnlichen Arrays möglich ist.
- Implementieren Sie weitere sinnvolle Methoden und Operatoren wie beispielsweise eine Methode `Loeschen()`, um Elemente zu löschen.

Beispiel: DAS IST EIN TEST

Verschlüsselung: **32 15 44 62 22 44 45 62 23 22 24 62 45 23 44 45**

Die Realisierung der Verschlüsselung setzt eine eingehende Auseinandersetzung mit den ein- und mehrdimensionalen Arrays aus dem Informationsteil voraus.

Durchführung:
Implementieren Sie ein Modul mit geeigneten Funktionen, die die o. a. Verschlüsselung durchführen können. Die verschlüsselten Zeichenketten sollen dabei in Arrays vom Datentyp `int` gespeichert werden. Das Modul soll für beliebige Schlüsselworte sowohl verschlüsseln als auch entschlüsseln. Beachten Sie bei der Umsetzung die Regeln der modularen Programmgestaltung. Trennen Sie Schnittstelle und Implementation.

Kontrolle:
Jedes Entwicklerteam stellt handschriftlich eine Verschlüsselungsmatrix mit einem selbst gewählten Schlüsselwort auf. Diese Matrizen dienen dann als Testgrundlage für die Kontrolle der korrekten Ver- und Entschlüsselung.

Testen Sie das Modul unter den Bedingungen des Auftrages. Verschlüsseln und entschlüsseln Sie sehr lange Zeichenketten, die vom Umfang her einem Memotext entsprechen. Das könnten ca. 300 Worte oder ca. 2000 Zeichen sein.

Lernziele:

- Sie lernen eine interessante Anwendung der Programmierung kennen – die Verschlüsselungstechnik.
- Sie erarbeiten die nötigen Kenntnisse über ein- und mehrdimensionale Arrays in C++.
- Sie erkennen die Besonderheiten der Zeichenkettenverarbeitung in C++.

Lernsituation 4:

Entwicklung eines Informationstools für die Anzeige von Umgebungsvariablen

Ausgangssituation:
Die Entwickler der Firma **ProSource** arbeiten oft mit den sogenannten Umgebungsvariablen, die das Betriebssystem (Windows oder UNIX) anbietet. In diesen Umgebungsvariablen stehen relevante Daten wie der aktuelle Username, der am System angemeldet ist, oder freigegebene Pfadangaben.

Bisher müssen die Entwickler die Informationen über einen Aufruf der Konsole und den entsprechenden Befehl (z.B.: SET unter Windows) abfragen. Diese Vorgehensweise ist mühsam und zeitaufwändig.

Aus diesem Grund hat das Entwicklerteam den Wunsch geäußert, dass die Firma ein Tool zur Verfügung stellt, das das Auslesen der Variablen angenehmer und schneller gestaltet.

Als Auszubildender der Firma **ProSource** erhalten Sie den Auftrag, dieses Tool zu programmieren. Das Tool sollte menügesteuert sein und verschiedene Filter für die Anzeige der Umgebungsvariablen anbieten.