

## Erste Schritte

Dieser Teil der Veranstaltung gibt einen ersten Eindruck der Programmierung mit C++. Es wird ein erstes Gefühl von Programmiersprachen vermittelt, ohne auf die gezeigten Bestandteile genau einzugehen; sie werden im späteren Verlauf der Vorlesung genau erklärt.

Zum Abschluss wird gezeigt, wie aus einem C++ Quell-Programm ein ausführbares Programm erzeugt wird.

### Inhalt

1. Installation von NetBeans.....	2
2. Das erste Programm.....	3
3. Variablen, Rechnen und Ein-/Ausgabe.....	5
4. Kontrollstrukturen.....	6
5. Eine Funktion.....	8
6. Eine Klasse.....	10
7. Vererbung: eine abgeleitete Klasse.....	12
8. Entwicklungsumgebungen.....	13
8.1. Das Arbeiten mit Eclipse.....	14

## 1. Installation von NetBeans

NetBeans ist ein komfortables Frontend für eine Menge von Werkzeugen (toolchain), wie Compiler, Debugger und Analysetools. eclipse benötigt eine so genannte Toolchain, damit man damit dann Programme erstellen kann.

Unix-basierte Betriebssysteme haben diese Werkzeuge „an Bord“. Für Windows muss man das zunächst installieren.

Gehen Sie wie folgt (Windows: 1-3, Unix: 4) vor, um NetBeans mit allen Werkzeugen zu installieren:

1. Installieren Sie cygwin ([www.cygwin.com](http://www.cygwin.com)) mit allen Entwickleroptionen.
2. Erweitern Sie den Systempfad von Windows um das bin-Verzeichnis von cygwin.
3. Jetzt sollte man aus einem Konsolfenster heraus  
c: gcc  
aufrufen können. Dann war die Installation erfolgreich.
4. Nun können Sie NetBeans herunterladen (<http://NetBeans.org/>) und installieren.

Ich **erwarte**, dass dies auf Ihren Rechnern bis **zur nächsten Vorlesung** versucht wurde.

Treten unerwartet Problem auf, demonstriere ich den Installationsprozess in der nächsten Vorlesung.

## 2. Das erste Programm

In dem Klassiker „Programmieren in C“ (Kernigham/Ritchi) wurde als erstes C-Programm das „Hello, World“ Programm vorgestellt. Es ist zum Standard für alle ersten Programme in Programmierveranstaltungen geworden:

```
$ cat hello.cpp
#include <iostream.h>           // stellt cout zur Verfügung
void main ()                   // vorgeschriebener Hauptprogrammname
{
    cout << "Hello, World" << endl; // Text ausgeben mit Zeilenende
}
$
```

Ein Aufruf des Programms bewirkt:

```
$ hello
Hello, World
$
```

## Bemerkung:

Die Beispielprogramme im Skript sind unter Unix erstellt und die Ausführung als Konsol-anwendung wird aus Sicht der Unix Shell gezeigt.

Unter Windows würde der Aufruf innerhalb einer DOS-Box wie folgt aussehen:

```
C:> hello  
Hello, World  
C:>
```

### 3. Variablen, Rechnen und Ein-/Ausgabe

Quadratzahlen kann man mit dem folgenden Programm ermitteln.

```
$ cat quadrat.cpp
#include <iostream.h>
void main () {
    int n, quadrat;                                // Deklaration von Variablen

    cout << "Bitte eine natürliche Zahl eingeben: "; // Eingabedialog
    cin  >> n;

    quadrat = n * n;                               // Berechnung

    cout << "Gelesen wurde n = " << n << endl      // Ergebnis ausgeben
         << "Dann ist n * n = " << quadrat << endl;
}
$
```

```
$ quadrat
Bitte eine natürliche Zahl eingeben: 6
Gelesen wurde n = 6
Dann ist n * n = 36
$
```

## 4. Kontrollstrukturen

Durch Kontrollstrukturen können Aktionen z.B. bedingt oder wiederholt ausgeführt werden.

Das folgende Beispiel druckt eine Euro/DEM Umrechnungstabelle aus:

\$ eurdem	
EUR	DEM
0	0
100	195.583
200	391.166
300	586.749
400	782.332
500	977.915
600	1173.5
700	1369.08
800	1564.66
900	1760.25
1000	1955.83

\$

```

$ cat eurdem.cpp
#include <iostream.h>
#include <iomanip.h> // wg. setw
void main ()
{
    const float unten=0; // Deklaration von Konstanten
    const float oben=1000;
    const int schrittweite=100;

    float dem, eur; // Deklaration von Variablen
    eur = unten; // Initialisierung von Variablen

    cout << "EUR" << "\t" << "DEM" << endl;
    while (eur <= oben) {
        dem = eur * 1.95583; // Berechnen
        cout << setw(5) << eur << "\t" << dem << endl; // Ausgeben
        eur = eur + schrittweite;
    }
}
$

```

## 5. Eine Funktion

Die Verwendung von Funktionen verdeutlicht das folgende Beispiel.

```
$ cat wurzel.cpp
#include <iostream.h>
float Wurzel(int x) // Funktionskopf
{ // Funktionsrumpf
    return sqrt(x);
}

void main ()
{
    int n; // Deklaration von Variablen
    float wurzel;
    cout << "Bitte eine natürliche Zahl eingeben: "; // Eingabedialog
    cin >> n;
    wurzel = Wurzel(n); // Berechnung
    cout << "Gelesen wurde n = " << n << endl // Ergebnis ausgeben
        << "Dann ist Wurzel(n) = " << wurzel << endl;
}
```

Vereinbarung

Verwendung

```
$ wurzel  
Bitte eine natürliche Zahl eingeben: 2  
Gelesen wurde n = 2  
Dann ist Wurzel(n) = 1.41421  
$
```

## 6. Eine Klasse

Eine Klasse ist eine Zusammenfassung von Datenstrukturen und Funktionen zur Manipulation dieser Strukturen und zu ihrem Zugriff.

Eine Uhr, die gestellt und die Zeit angezeigt werden kann, kann (ohne ticken) beschrieben werden durch folgendes Programm.

Ein Aufruf soll bewirken:

```
$ uhr  
17:32  
$
```

```
$ cat uhr.cpp
#include <iostream.h>
class Uhr { // Definition einer Klasse, bestehend aus:
    int Stunde; // Attributen und ...
    int Minute;
public:
    void ZeitAnzeigen (); // ... Prototypen von Elementfunktionen
    void Stellen (int stunde, int minute);
};
```

```
void Uhr::ZeitAnzeigen () // Definition der Elementfunktionen
{ /* hierhin gehört, was die Funktion tut */
    cout << Stunde << ":" << Minute << endl;
}
```

Vereinbarung

```
void Uhr::Stellen (int stunde, int minute)
{
    Stunde = stunde;
    Minute = minute;
}
```

```
void main ()
{ Uhr MeineWanduhr; // Deklaration eines Objekts der Klasse
  MeineWanduhr.Stellen (17, 32); // Aufruf einer Funktion des Objekts
  MeineWanduhr.ZeitAnzeigen (); // Aufruf einer anderen Funkt. des Objekts
}
```

Verwendung

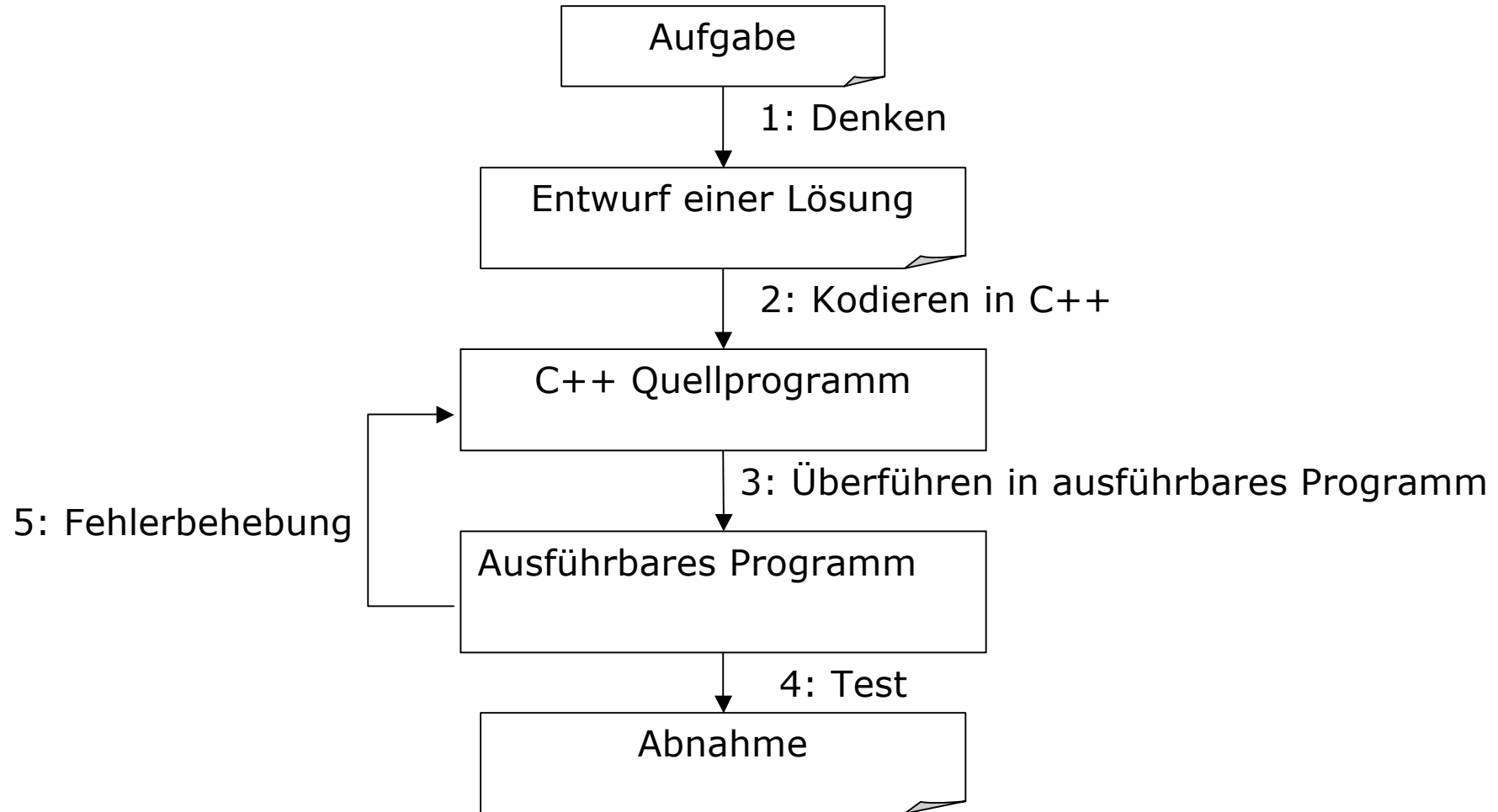
## 7. Vererbung: eine abgeleitete Klasse

```
class Wecker : public Uhr { // Wecker wird abgeleitet von Uhr
    int WeckStunde;        // hat zusätzliche Attribute und ...
    int WeckMinute;
public:
    bool Alarm ();        // ... zusätzliche Elementfunktionen
    void WeckzeitEinstellen (int stunde, int minute);
};
bool Wecker::Alarm ()
{
    /*...*/
}
void Wecker::WeckzeitEinstellen (int stunde, int minute)
{
    /*...*/
}

void main ()
{
    Wecker MeinWecker;    // Deklaration eines Wecker-Objekts
    MeinWecker.Stellen (17, 32); // Aufruf einer geerbten Funktion
    MeinWecker.WeckzeitEinstellen (6, 30);
    if (MeinWecker.Alarm ())
        cout << "Aufstehen !";
}
```

## 8. Entwicklungsumgebungen

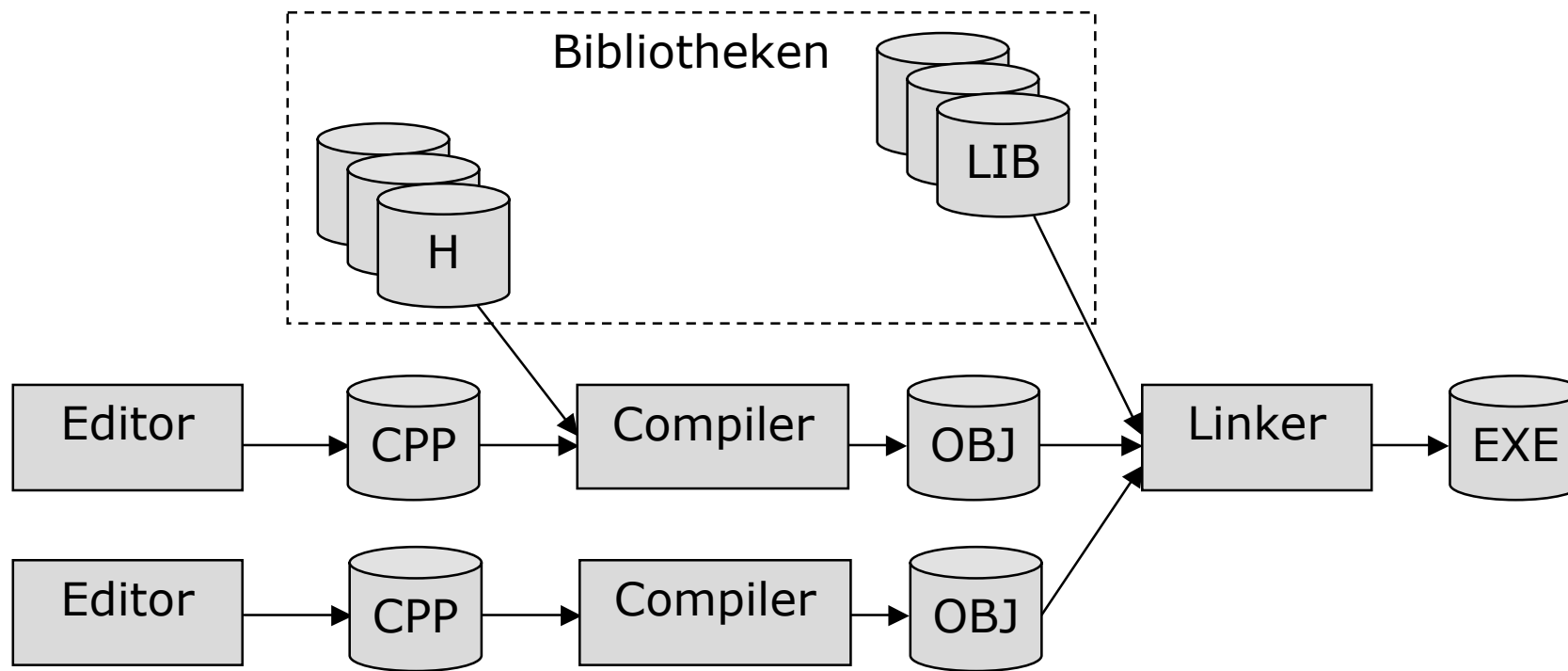
Die Entwicklung von Anwendungen erfolgt in mehreren Schritten:



Jeder Schritt hat typische Aktivitäten, wobei die Schritte 2-5 durch Werkzeuge unterstützt werden.

Für Kodieren, Übersetzen und Testen wird im Praktikum Quellen: Unix-Shell mit GNU C++ Übersetzer und Debugger: [www.cygwin.com](http://www.cygwin.com)

Den Prozess der Erstellung eines ausführbaren Programm, ausgehend von einem C++ Quellprogramm kann man wie folgt verdeutlichen:



## 8.1. Das Arbeiten mit Eclipse

-> vi, make, eclipse am System