



Einführung in die Programmierung mit *Delphi*

Tobias Giese

19.12.2006

```
program MeinProgramm
uses SysUtils;
var a, b, ergebnis : Integer;
begin
    if ( a + b > 0 ) then
        ergebnis := sqrt( a + b );
end.
```

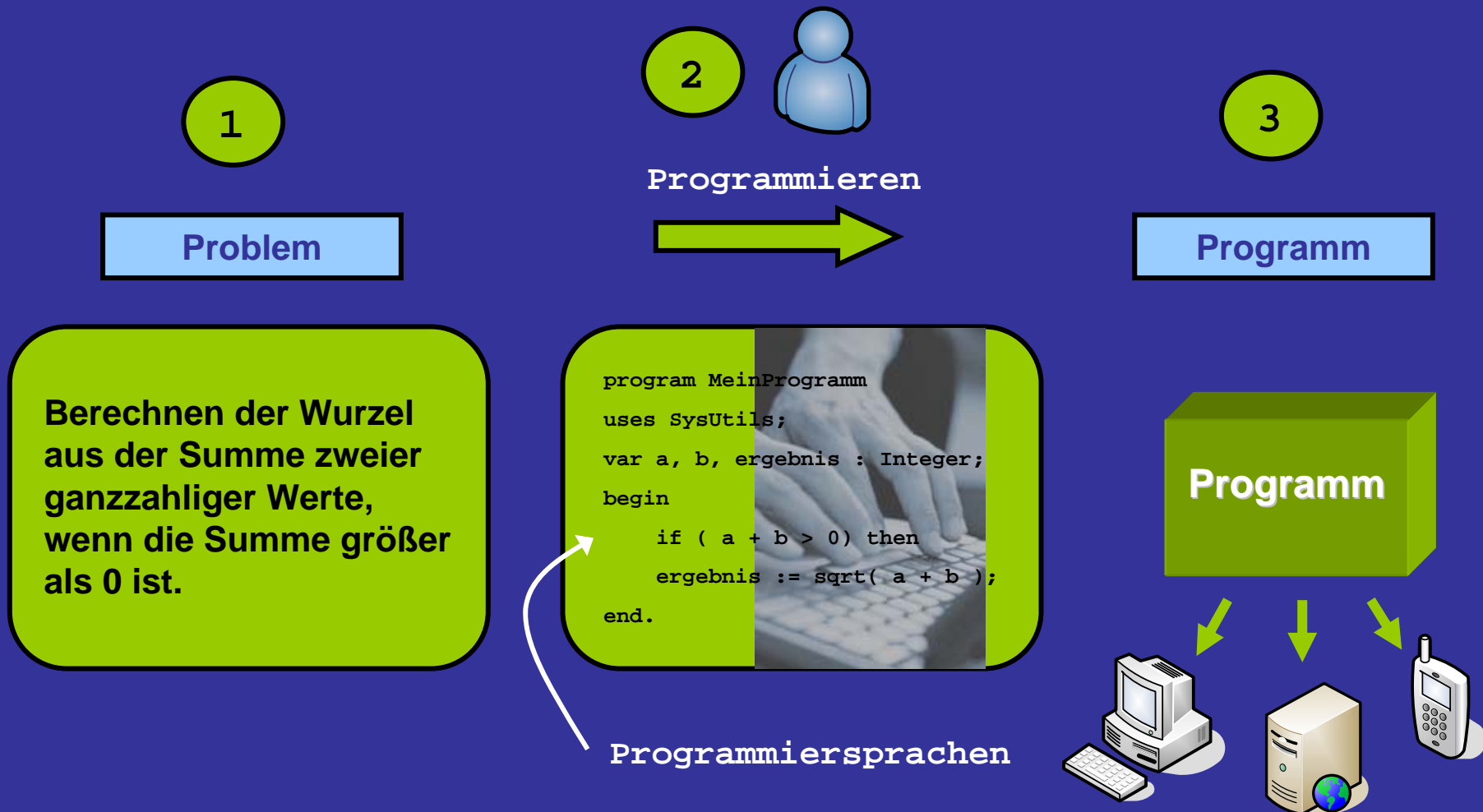
Die heutigen Themen im Überblick

◆ Agenda

- Grundbegriffe
- Geschichte der Programmiersprachen
- Entwicklungsprozess eines Programms
- Einführung in Delphi
- Programmstrukturen

Was ist Programmieren?

◆ Ausgangssituation



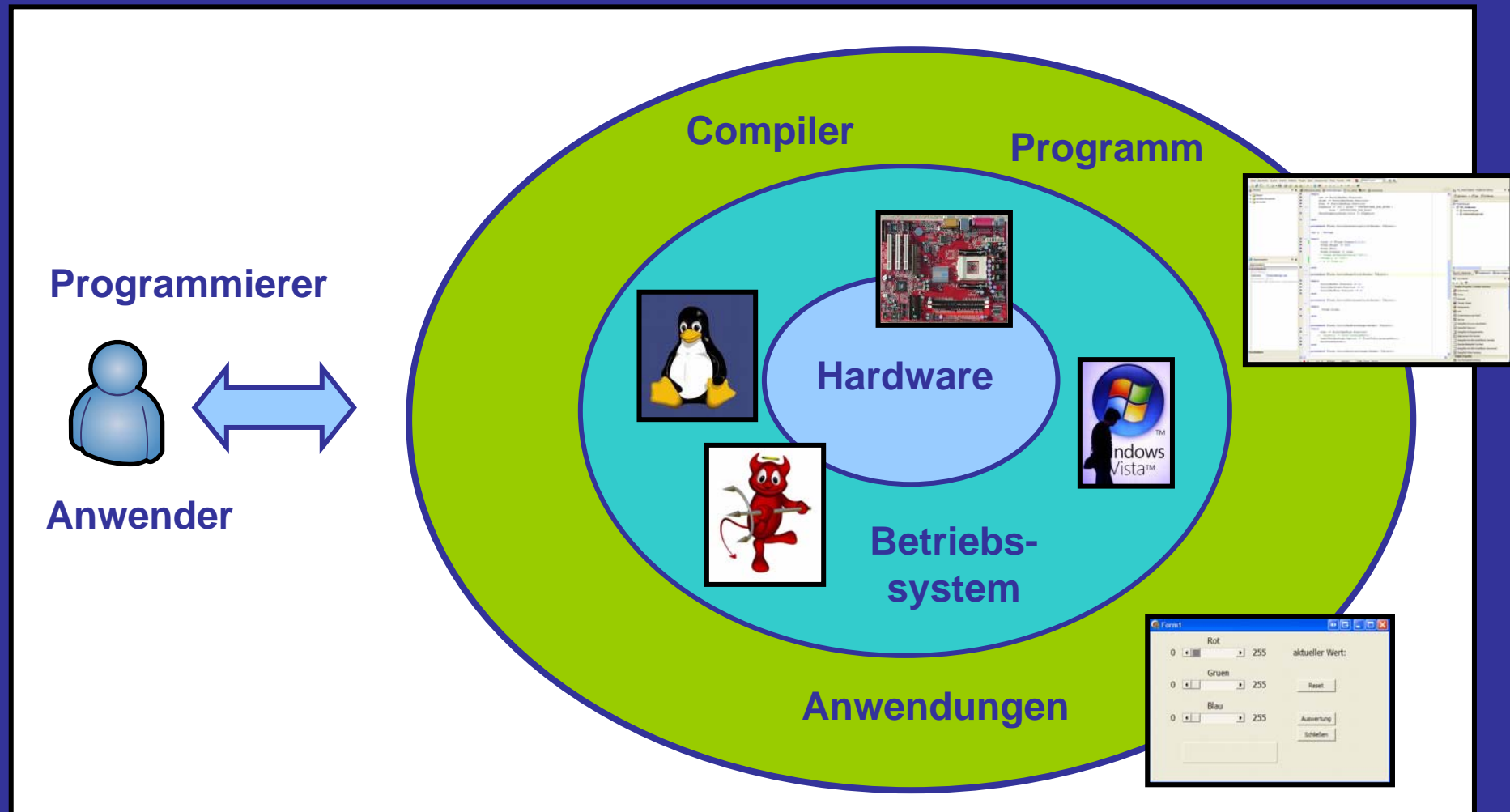
Grundbegriffe

◆ Programme und Programmiersprachen

- Programmiersprache
 - eine zum Abfassen von **Computerprogrammen** geschaffene **Sprache**
- Programm
 - eine vom Computer interpretierbare vollständige Arbeitsanweisung zur **Lösung** einer Aufgabe
- Programmierumgebung
 - eine Anwendung die beim Entwickeln von Programmen hilft
- Prozess
 - ein Programm in Ausführung
 - wird vom **Betriebssystem** verwaltet
 - ausgeführt vom **Prozessor**

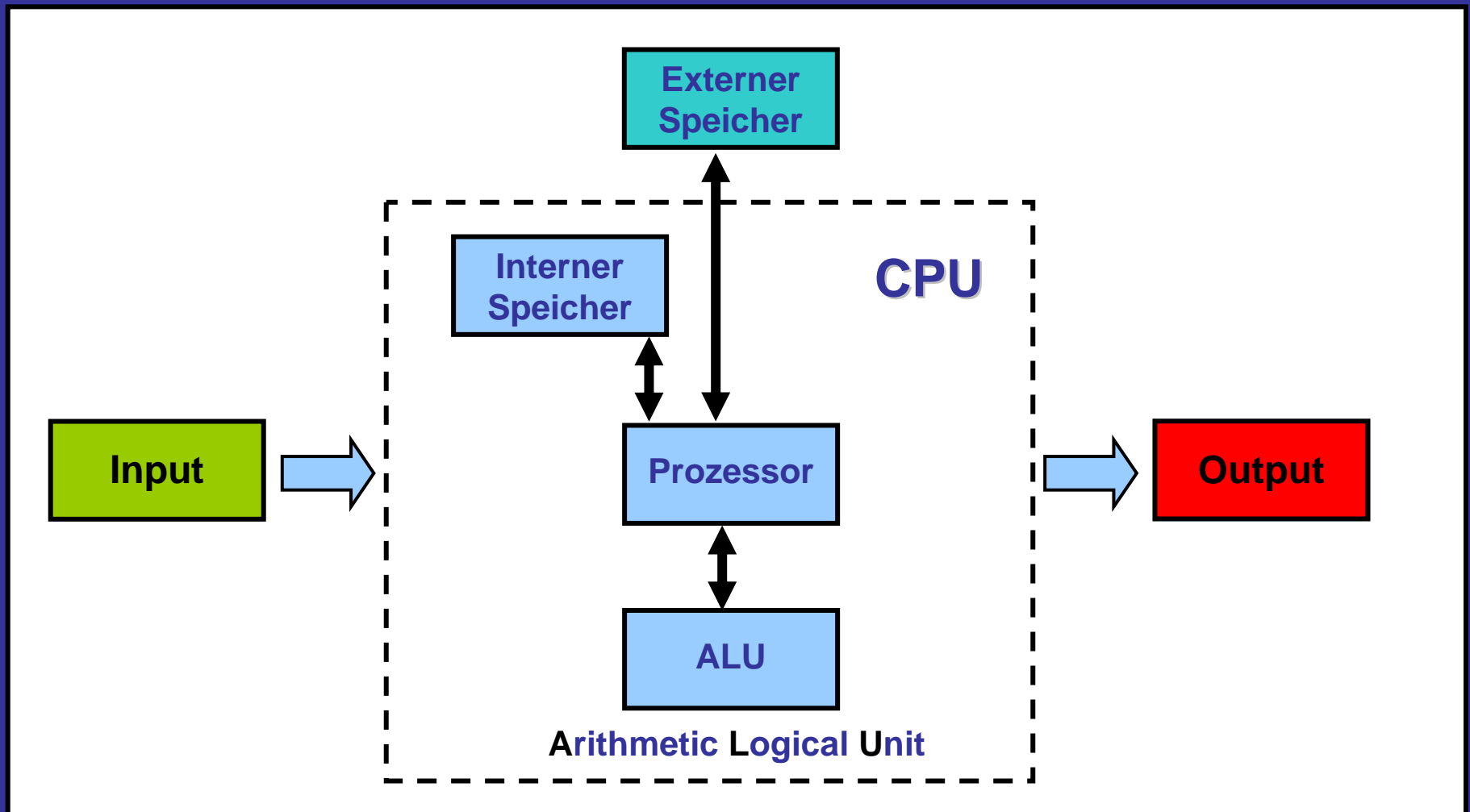
Schalenmodell

◆ Anwendung – Betriebssystem – Hardware



Das Herzstück eines Computers

◆ Central Processing Unit (CPU)



1. Generation – der Anfang

◆ Generationen der Programmiersprachen

1. Generation – Binäre **Maschinensprache**

- Computer Hardware verarbeitet Folgen von Nullen und Einsen
- Programmierer muss Steuercode schreiben und Speicherverwaltung selbst regeln
- bedeutet Einlesen der Daten in CPU und Ergebnisdaten ausschreiben passiert alles im Binärformat!
 - z.B.: Addieren-Steuerbefehl

reale Aufgabe:

3 + 4



Maschinensprache:

00011010

0011

0100

Operation „addiere“

3

4



Probleme:

- Befehle abhängig vom jeweiligen Prozessor
- Programme sehr zeitaufwendig zu schreiben und schlecht zu warten

2. Generation – erste Vereinfachungen

◆ Generationen der Programmiersprachen

2. Generation – **Assembler**

- Steuerbefehle sind nun kleine Kürzel aus meist 3 Buchstaben
 - z.B.: Addieren-Steuerbefehl
- Speicherverwaltung noch Aufgabe des Programmierers
- Umwandlung von Assemblercode in Maschinensprache durch Assembler

reale Aufgabe:

3 + 4



Assemblersprache:

ADD

AX

BX

Operation „addiere“

Register mit 3

Register mit 4



Probleme:

- Befehle abhängig vom jeweiligen Prozessor
- Programme sehr zeitaufwendig zu schreiben und schlecht zu warten

3. Generation – das Lernen wir

◆ Generationen der Programmiersprachen

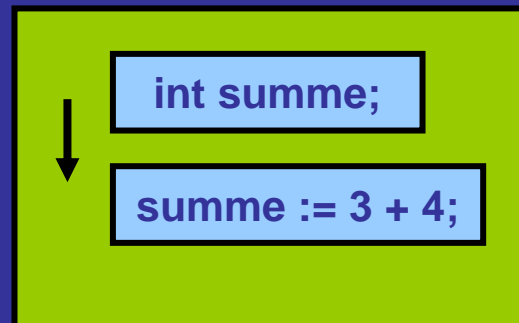
3. Generation – Hochsprachen



- Unabhängigkeit der Programmiersprache vom verwendeten Computer
- Übersetzung des Programms in Maschinensprache per **Compiler** (vor der Ausführung) oder **Interpreter** (während der Ausführung)

reale Aufgabe:

3 + 4



Variable festlegen

Addieren und
zuweisen des
Ergebnisses

Programmiersprachen der 3.Generation:

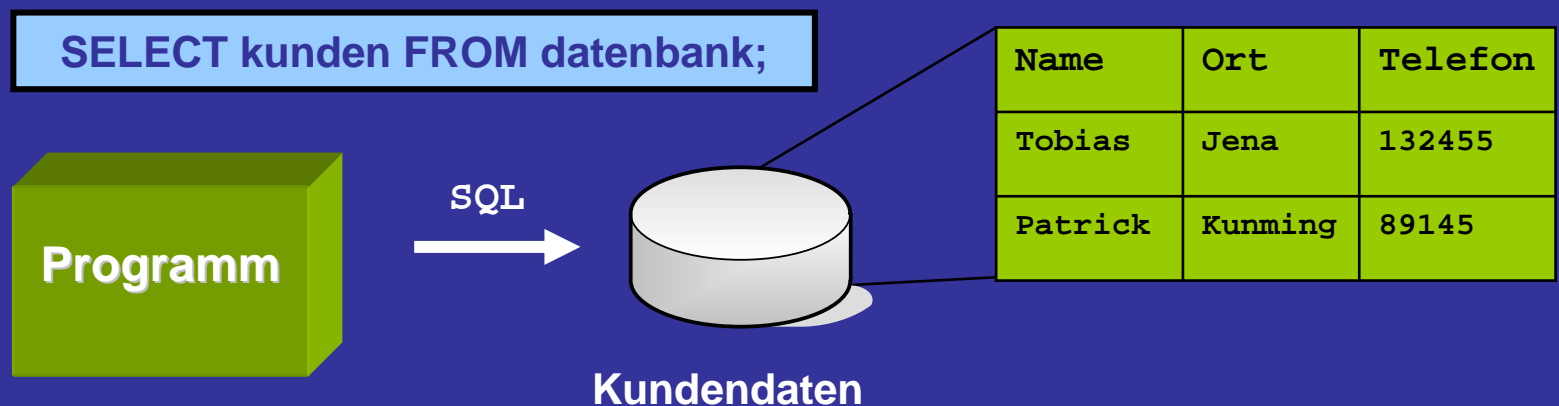
- Java, C++, Pascal, Delphi

4. Generation – komplexe Sachverhalte

◆ Generationen der Programmiersprachen

4. Generation – **Abfragesprachen**

- bieten einfache Sprachmittel zur Auslösung komplexer Operationen
- es wird nur die Frage formuliert, aber nicht der Weg
- es soll das gleiche wie mit Drittgenerationssprachen erreicht werden, allerdings mit weniger Programmcode
- z.B.: Datenbankabfragen

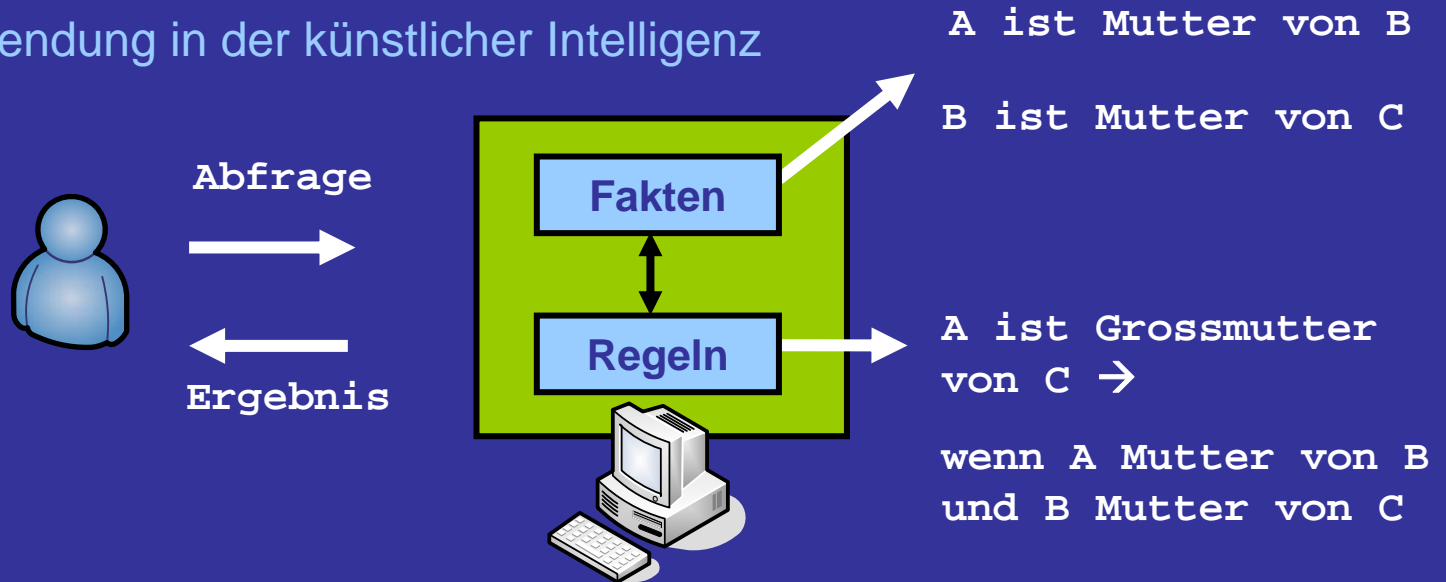


5. Generation – künstliche Intelligenz

◆ Generationen der Programmiersprachen

5. Generation – **Very High Level Language**

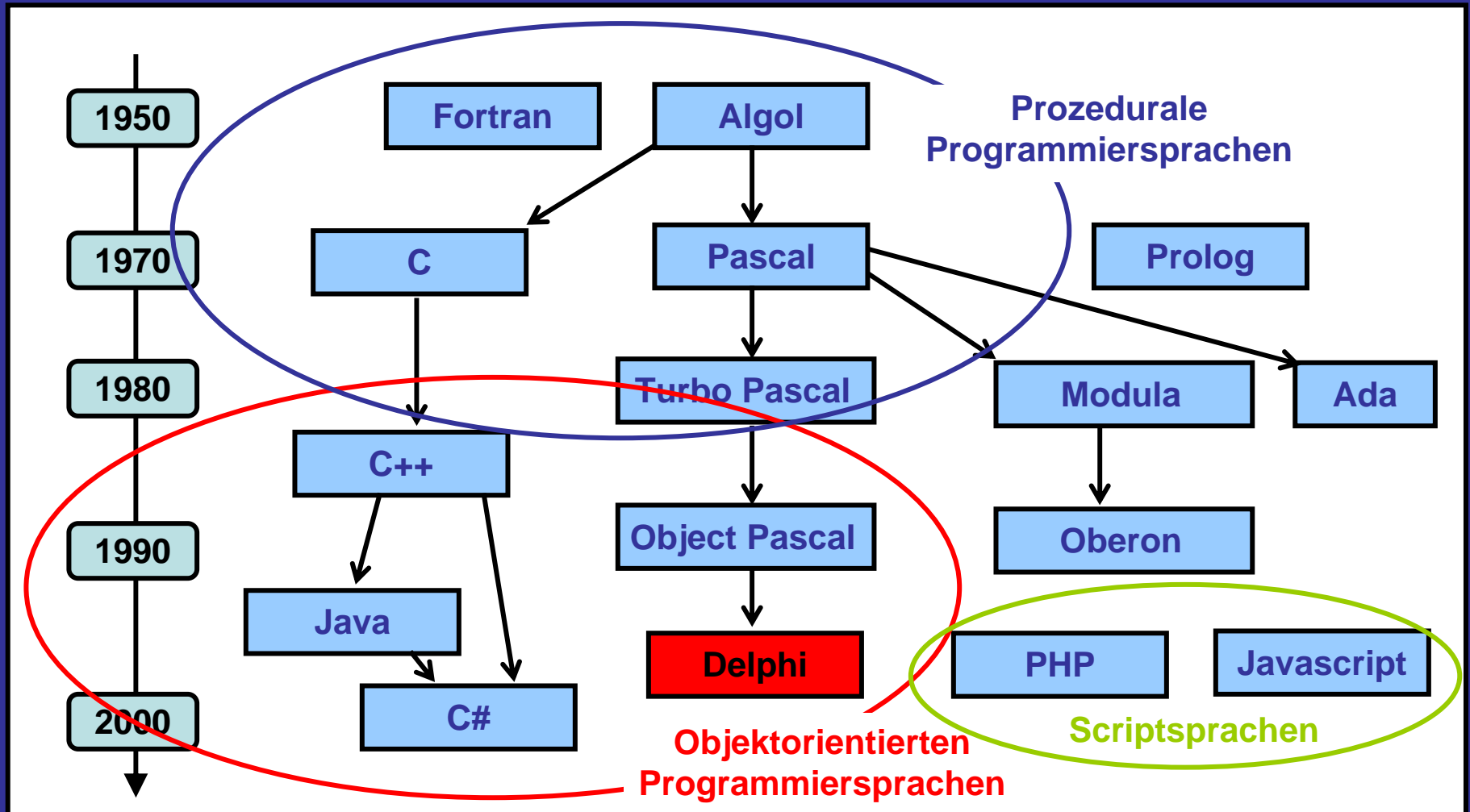
- Beschreiben Sachverhalte und Probleme
- Anwendung in der künstlicher Intelligenz



Frage: Ist A die Grossmutter von C?

Antwort: **yes**.

Geschichte der höheren Programmiersprachen



Paradigma : wissenschaftliche Denk- oder Sichtweise

◆ Programmierparadigmen – Prinzipien

Imperative
Ansatz



Deklarative
Ansatz

Programm besteht aus
Anweisungsschritte die ein
Problem lösen!

Wie und in welchen
Reihenfolge wird ein Problem
gelöst?

Algorithmus:

genaue Handlungsvorschrift
zur Lösung eines Problems

Was gilt?

→ Problemlösung wird aus
Regeln durch Computer
hergeleitet indem eine Frage
gestellt wird.



Logische
Programmierung



Funktionale
Programmierung

◆ Programmierparadigmen

**Imperative
Ansatz**



**prozedurale
Programmierung**



**objektorientierte
Programmierung**

**Programm besteht aus
Anweisungsschritten, die das
Problem lösen**

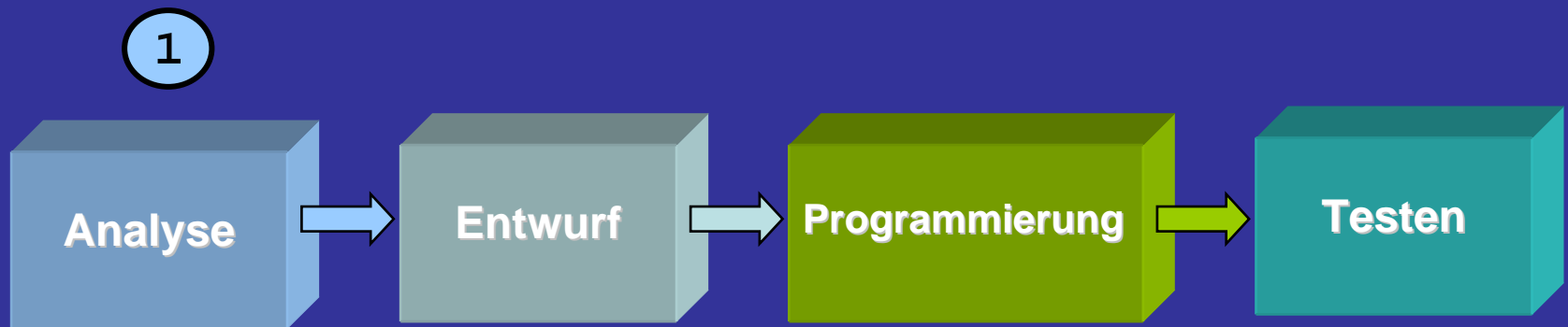
**Kern ist der Algorithmus,
der das Problem löst**

**Kern ist das Objekt, das Daten
und Verhalten enthält**

**→ Algorithmen werden zum Teil
des Objektes**

Zu Beginn einer Programmentwicklung

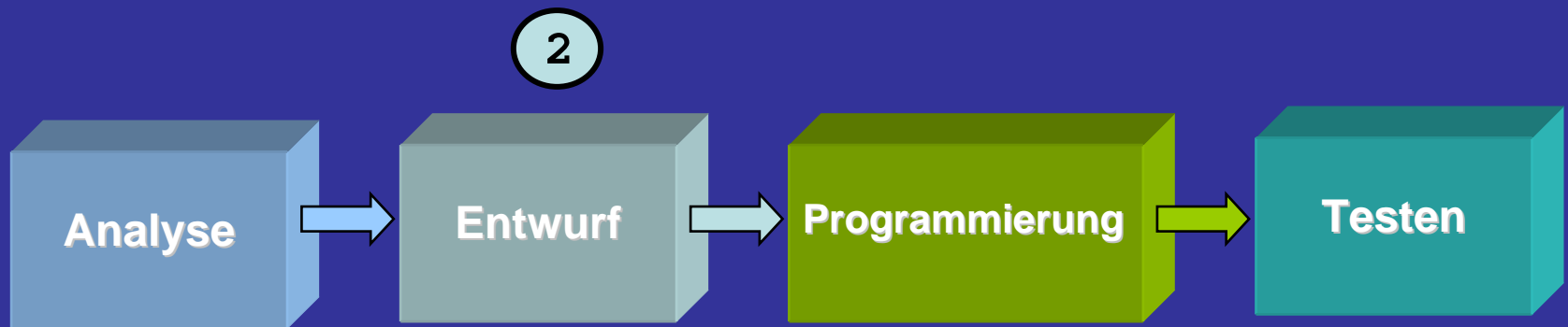
◆ Softwareentwicklungsprozess



- **Anforderungen erheben**
→ was soll mein Programm können?
- **Grobkonzept der Software**
- **Projektplan (Ressourcen, Timeline)**
→ Plan um das Programm zu realisieren

Wir entwerfen das Programm

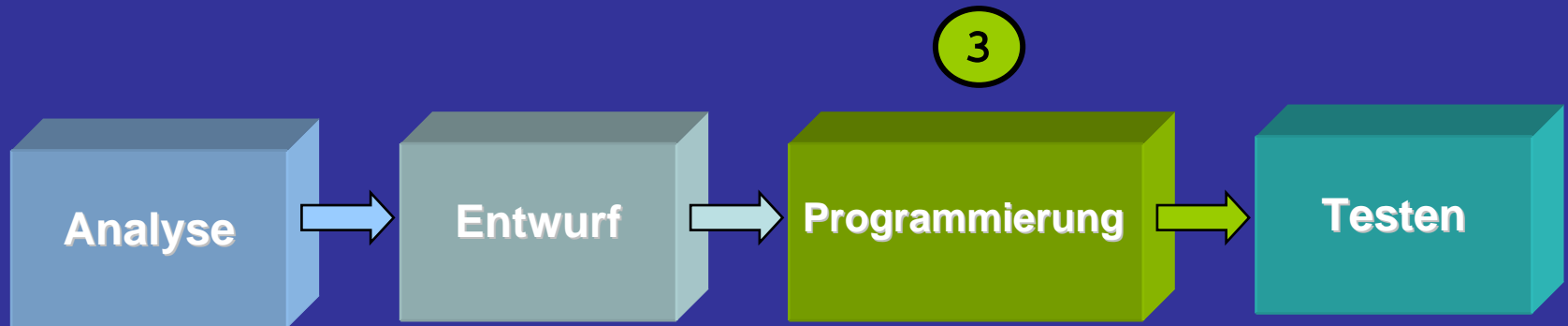
◆ Softwareentwicklungsprozess



- **Software-Architektur**
- **technischer Entwurf (Abläufe, Strukturen)**
- **Lösungskonzept**

Was ist Programmieren?

◆ Softwareentwicklungsprozess



- Umsetzen des Entwurfs in ein konkretes Programm
- Entwickeln einzelner Module / Einheiten
- Kombination dieser Module zu einem funktionsfähigen Gesamtprogramm

Doch was bedeutet das für mich als Schüler ?



◆ Programmentwicklung – Teil 1

Problem-
formulierung



Problem-
analyse



Berechnen der Wurzel aus der Summe zweier ganzzahliger Werte, wenn die Summe größer als 0 ist.

Modularisierung → Zerlegung des Gesamtproblems in Teilprobleme

z.B.: Erst Summenbildung, dann Wurzelziehen

◆ Programmentwicklung – Teil 2

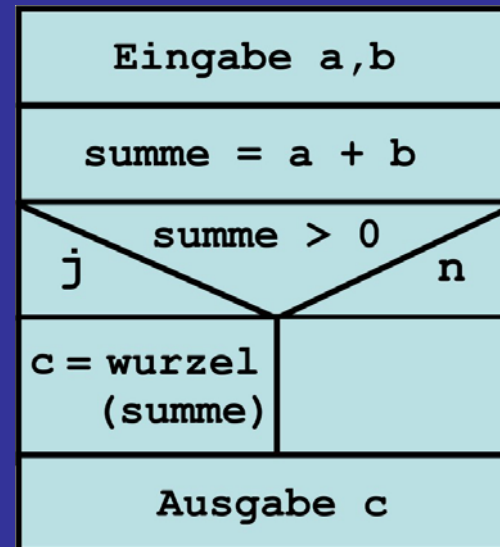
Entwurf der
Algorithmen



Programmieren
der Algorithmen



Testen des
Programms



Verwendung von:

- Struktogrammen
- Programmablaufplänen

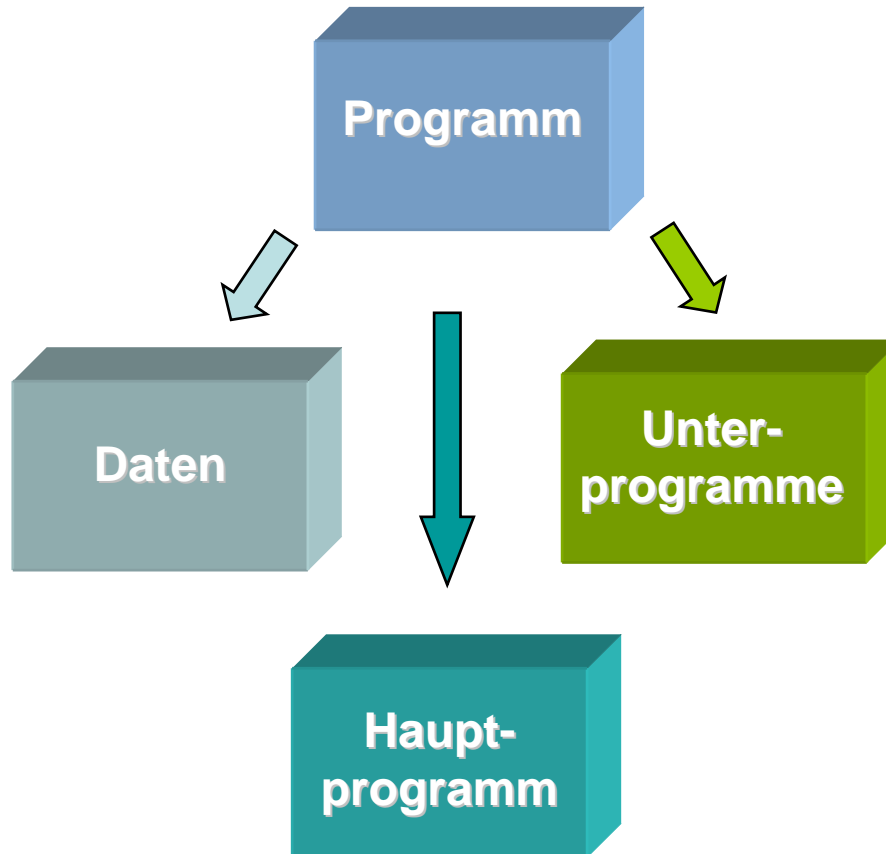
Fragen wir das Orakel

◆ Was ist Delphi ?

- objektorientierte Programmiersprache
- Delphi ist eine Programmierumgebung von Borland
- entstammt aus Pascal → Object Pascal → Delphi (seit 2003)
- einfach zu erlernen
- grosse Anzahl an mitgelieferten Oberflächenkomponenten zur Erstellung von graphischen Oberflächen
- grosse Community

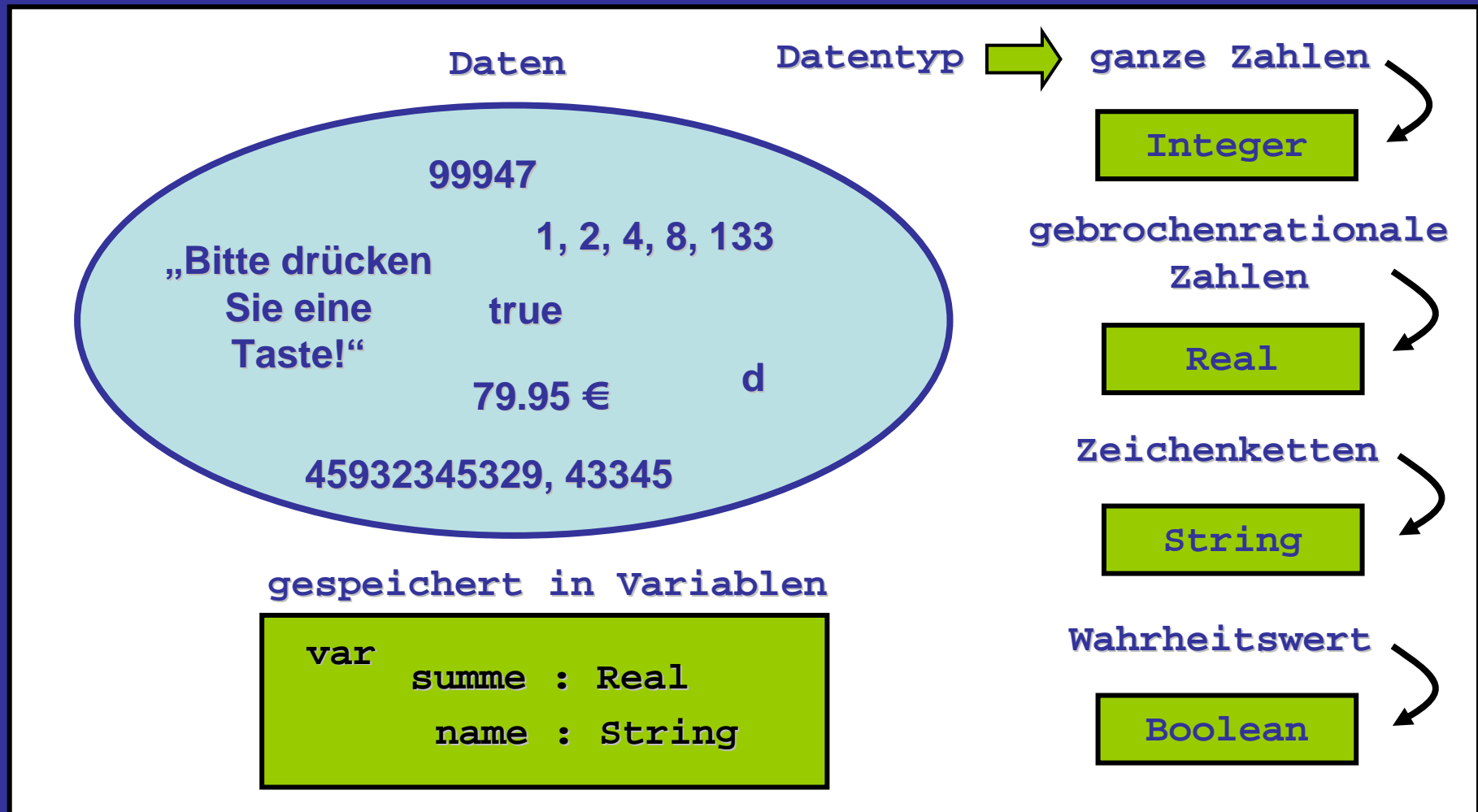
Woraus besteht ein Programm?

◆ Programmaufbau





◆ Daten und Datentypen

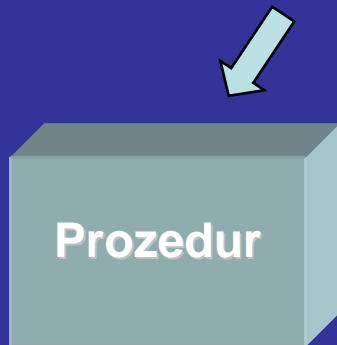




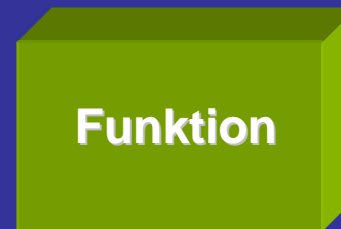
◆ Unterprogramme – beschreiben das Verhalten

- für sich abgeschlossener Programmteil
- wird meist aus Hauptprogramm aufgerufen
- enthalten Anweisungen die mehrmals verwendet werden
- es gibt zwei Formen in Delphi

- kein Rückgabewert



```
procedure summe();  
begin  
    summe := 3 + 4  
end;
```



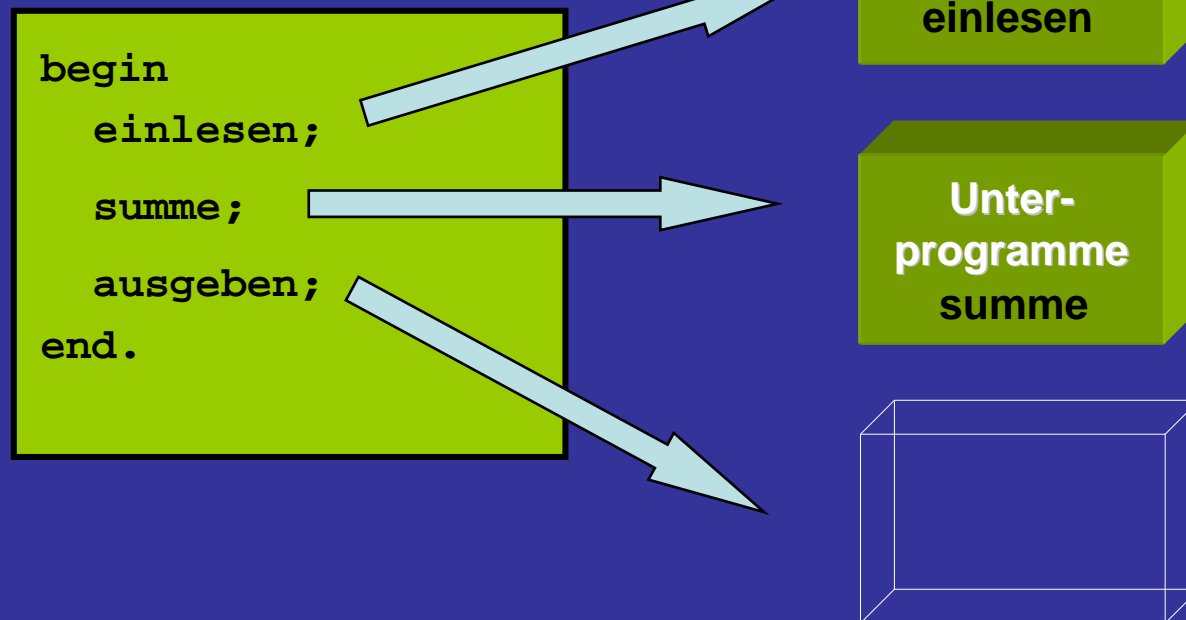
- Rückgabewert

```
function summe() : Real;  
begin  
    result := 3 + 4  
end;
```



◆ Hauptprogramm

- befindet sich am Ende des Programms
- üblicherweise sehr kurz
- enthält Aufrufe von Unterprogrammen



◆ Struktur eines Delphi – Programms

One-Pass-
Compiling



Program

Programmkopf

Konstanten

Datentypen

Variablen

Unterprogramme

Hauptprogramm

Programmcode

`program`

`const`

`type`

`var`

`procedure,`
`function`

`begin`
`end.`

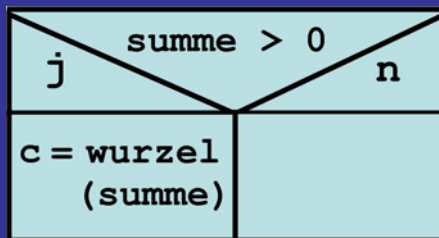
**Reihenfolge
wichtig !**

Beeinflussen des Programmablaufs

◆ Kontrollstrukturen – Flusskontrolle

- Entscheidung

Struktogramm



Programmcode

```
if (summe > 0) then
    c = wurzel(summe);
else
```

- Wiederholung
– Schleifen

```
while (zaehler < 5) do
    begin
        summe;
    end;
```

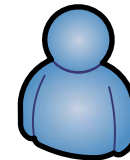
◆ Ein einfaches **Consolen** - Programm

**Programm
Eingabe**

**Programm
Ausgabe**

```
program MeinProgramm
{$APPTYPE Console}
uses
    SysUtils,
    Eingabe, Ausgabe;
begin
    { Hier Code einfügen }
end.
```

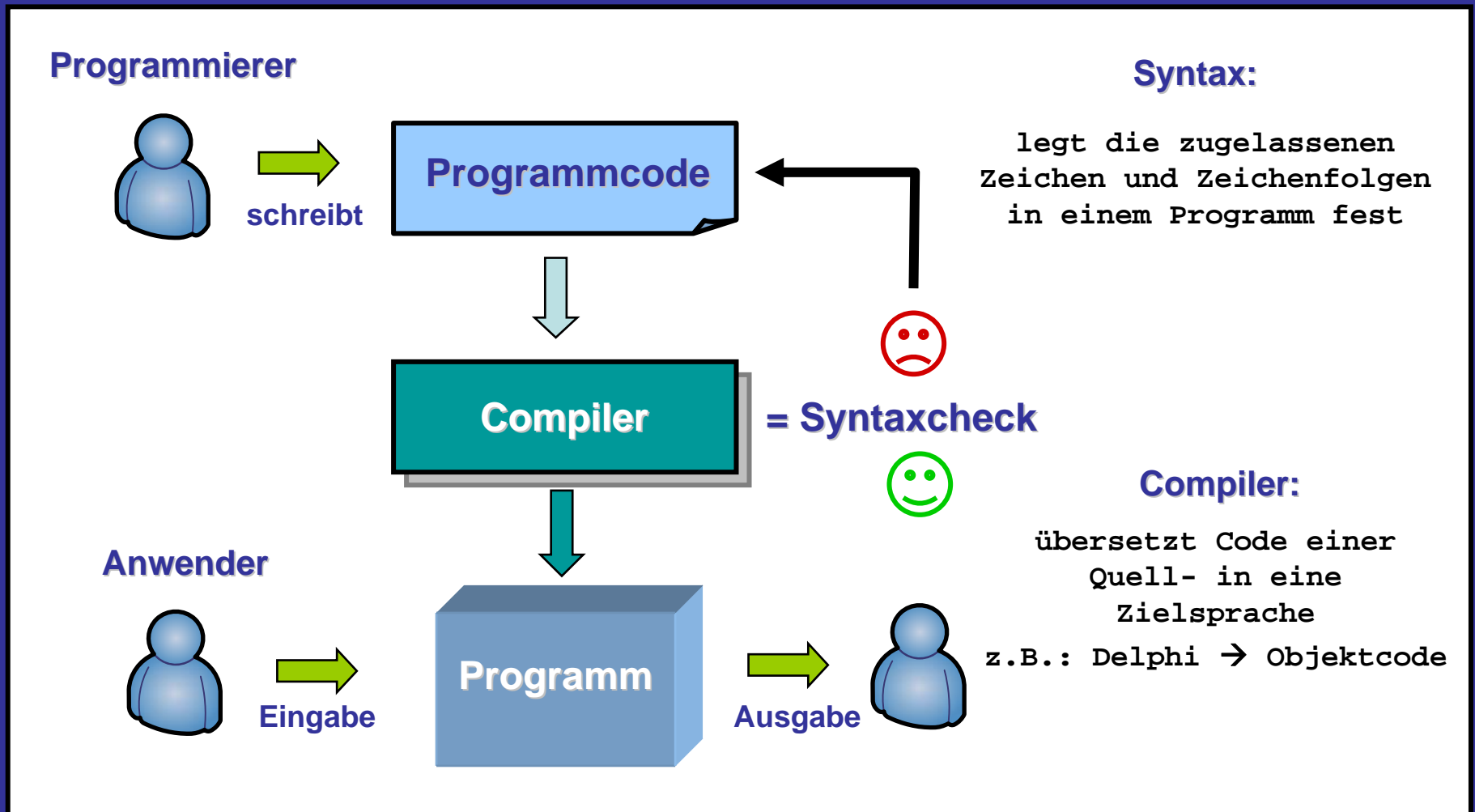
Programmierer



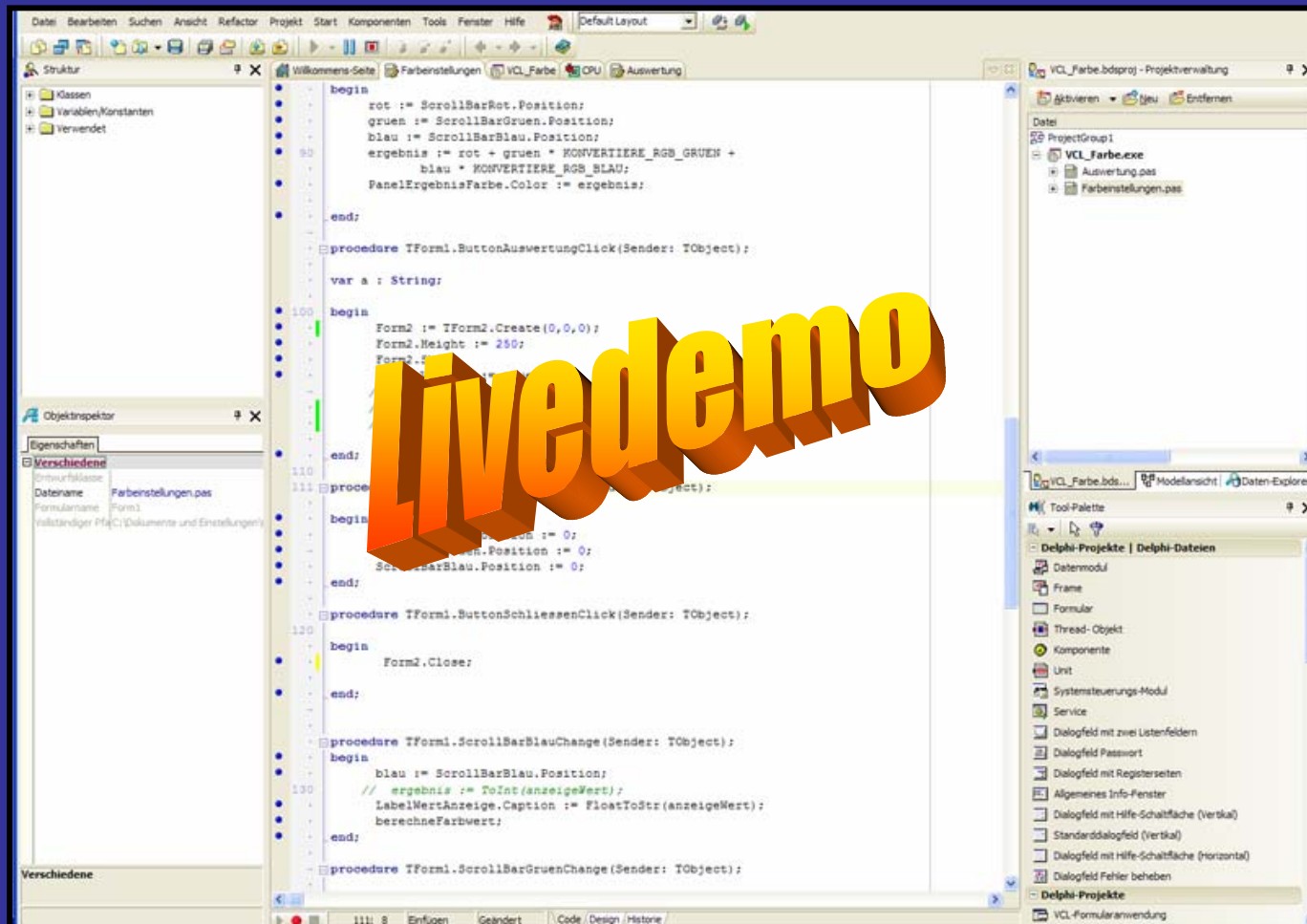
Algorithmus

Delphi ist case-insensitiv! z.B.: egal ob program oder Program oder PrOgrAm

◆ Programmcode → ausführbares Programm

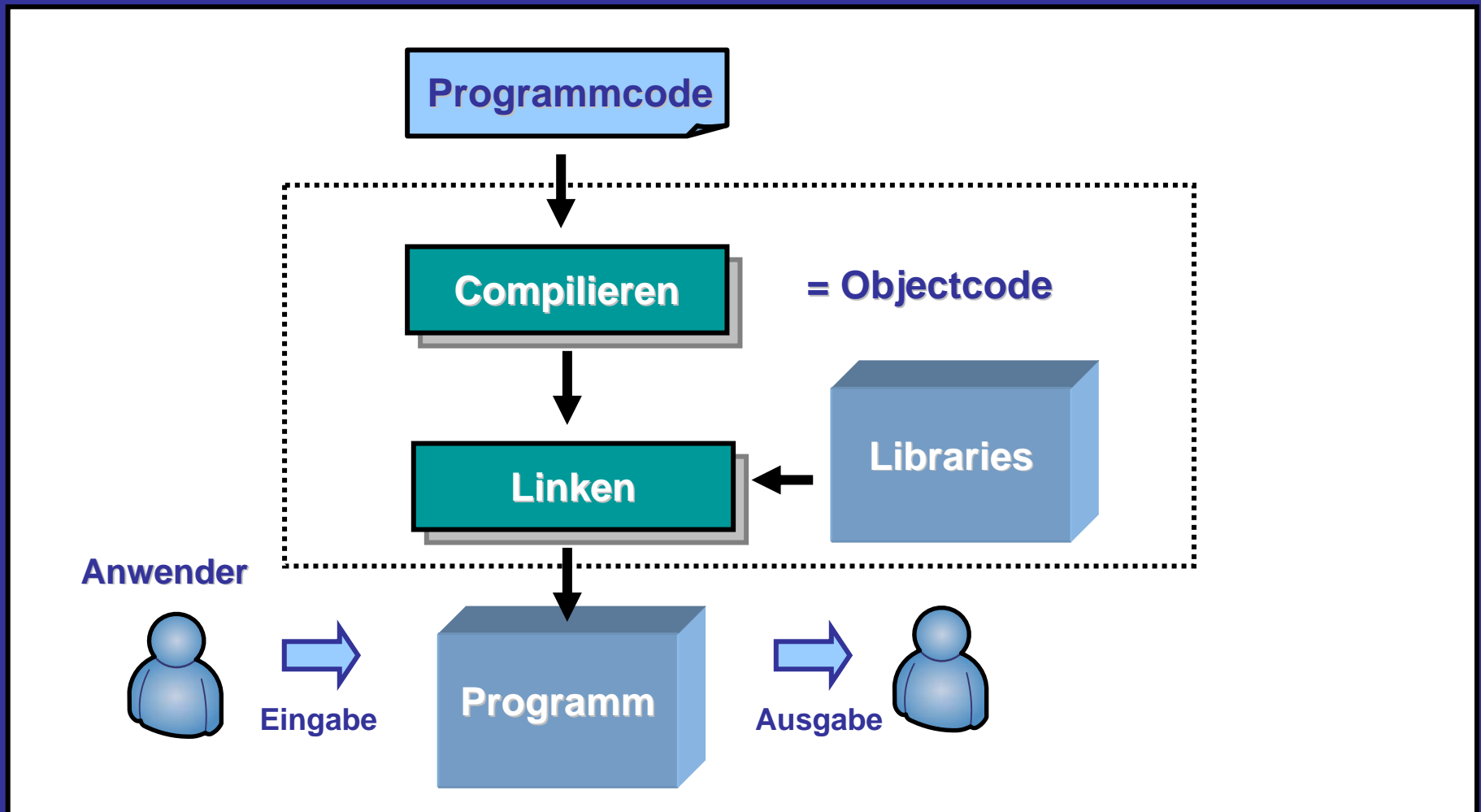


Praxisbeispiel



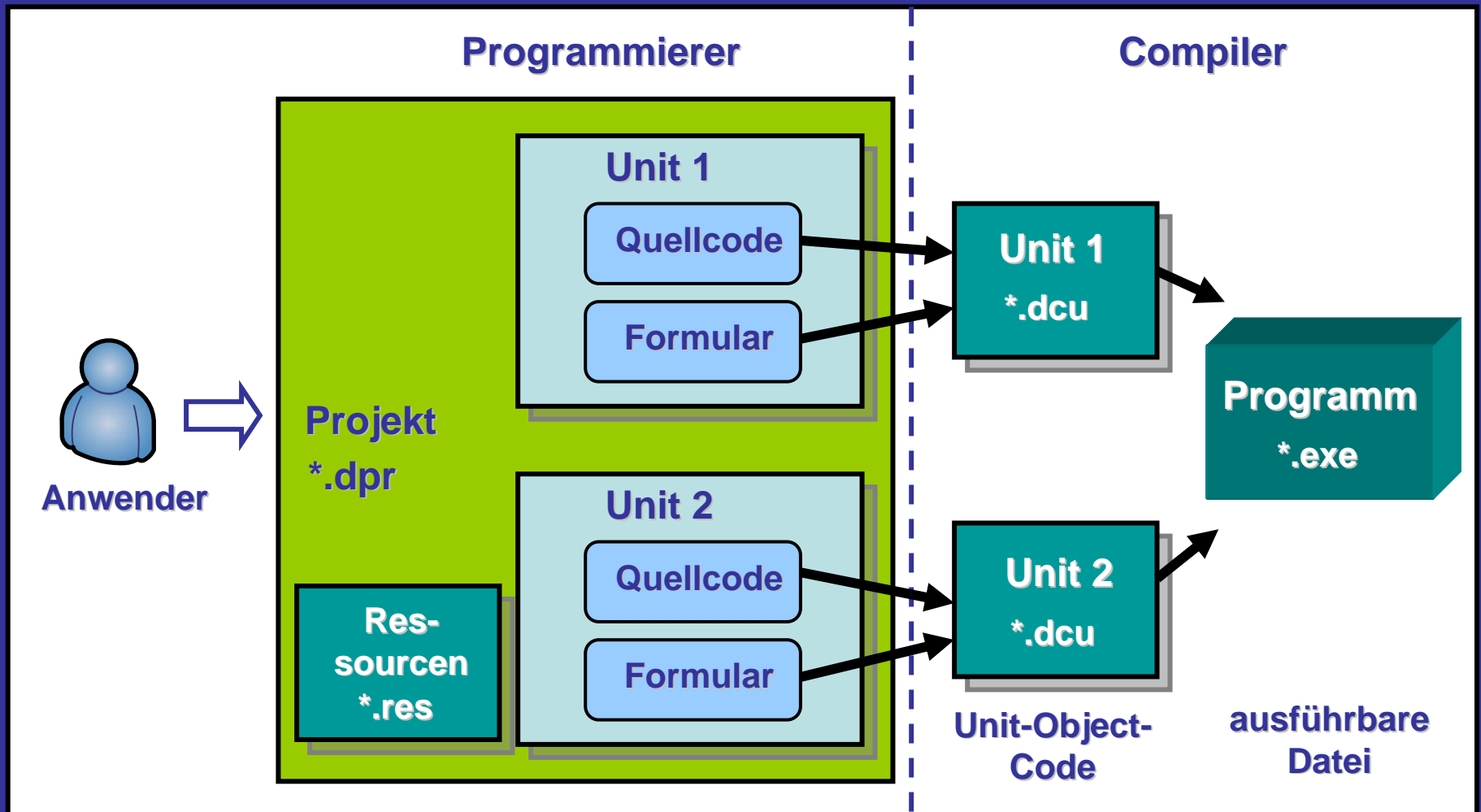
Weitere Infos – Nicht Teil des Vortrages

◆ Erstellung eines Programms – Überblick



Programm als Teil eines Projektes

◆ Erstellung eines Programms in Delphi

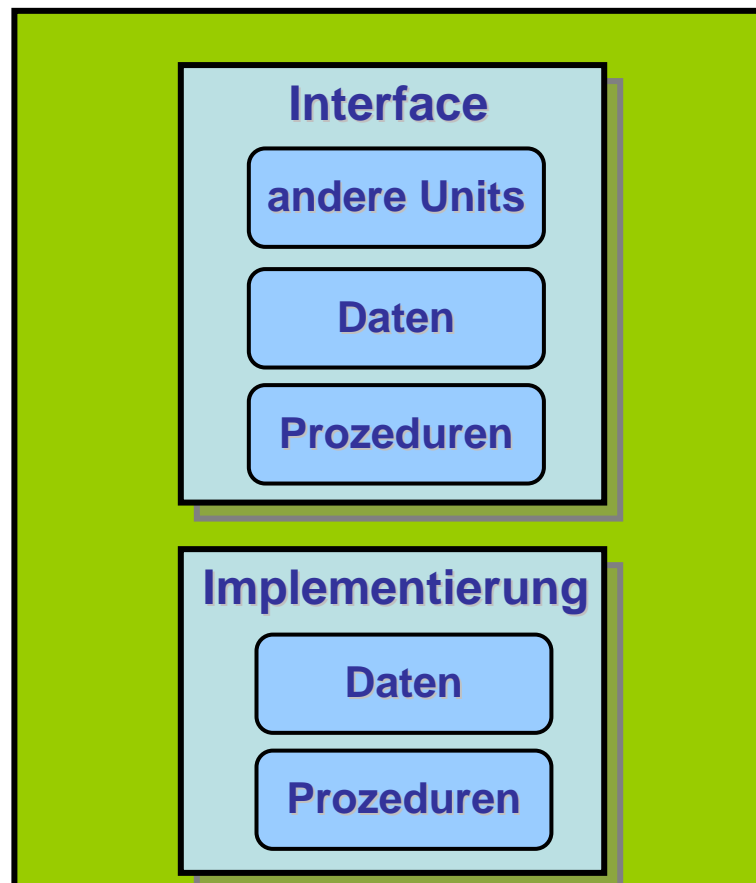


◆ Strukturen von Delphi

- Eigenschaften
 - angelehnt an reale Welt
 - geben den Zustand an → Attribute einer größeren Einheit
 - z.B.: Größe, Farbe, Anzahl
- Ereignisse
 - Programm kann auf externe Ereignisse reagieren
 - Mausklicks, Tastendrücken, das Öffnen/Speichern eines Dokumentes oder andere Programme lösen Ereignisse aus
- Methoden
 - beschreiben das Verhalten

◆ Implementierung vs. Interface

Programm / Unit / Library



Interface = Schnittstelle

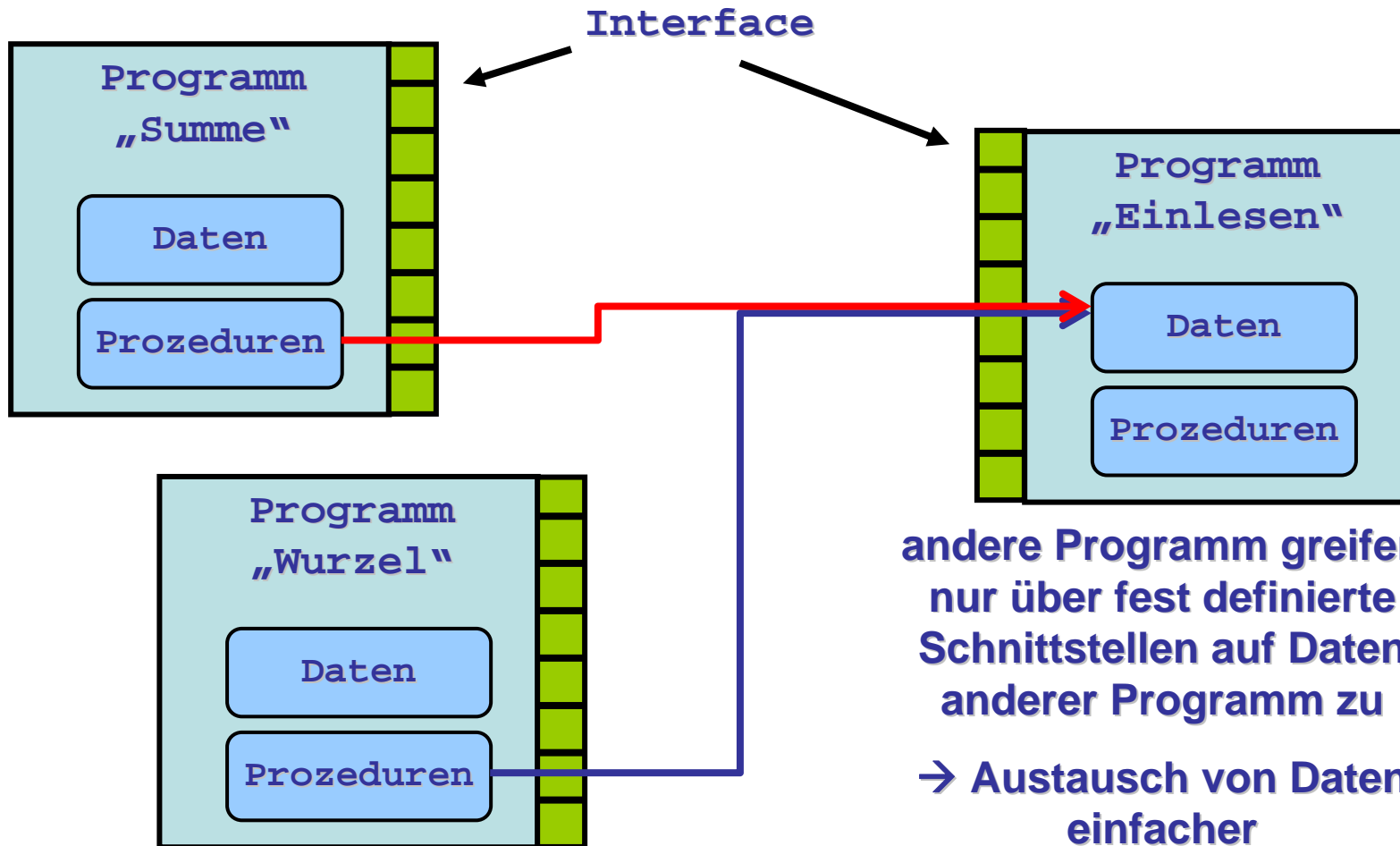
gibt an welche Methoden oder Attribute von „außen“ zugreifbar sind

Interface enthält keine Methodenrumpfe

Implementierung

enthält konkrete Algorithmen und Logik

◆ Übertragungsmedien



◆ Wissenswertes über Delphi

- Programmzeile kann beliebig lang sein
- Jede Programmzeile endet mit einem Strichpunkt
- Hauptprogramm: begin und end mit Punkt
- es dürfen Groß- und Kleinschreibung beliebig gemischt werden
 - → Delphi ist case-insensitiv
- Kommentare
 - // eine Zeile
 - { } mehrzeiliger Kommentar
 - (* *)
 - jeder Text nach finalem end wird ignoriert (vom Compiler)
- → Compiler Kommentare = Directives

◆ Geltungsbereich vs. Sichtbarkeit

- Geltungsbereich → Variablen, Konstanten und Typdeklarationen
 - gültig für das ganze Programm (Unit) → globale Gültigkeit
 - gültig für einen Abschnitt (Methode) → lokale Gültigkeit
- Sichtbarkeit → regelt den Zugriff von außerhalb
 - nur für globale Definitionen
 - public – protected – private

◆ Wissenswertes über Interfaces

- Hilfsfunktionen benötigen keine Interface Deklaration
- Anlegen eines neuen Formulars

```
Form2 := TForm2.Create(Parent);  
Form2.Height := 200;  
Form2.Show;  
Form2.Visible := true;
```

- Benutzen anderer Methoden aus anderen Units
- Typumwandlung (aus Unit SysUtils)
 - trunc → wandelt Real in Integer-Werte
 - StrToFloat → wandelt eine Zeichenkette in einen Real-Werte
 - FloatToStr → wandelt einen Real-Wert in Zeichenkette

◆ Klassenmethoden

- Methode die einer Klasse gehört und nicht einer speziellen Instanz (oder auch Objekt genannt)

```
type
  TgeomForm = class
    {...}
  public
    class function dimension : Integer;
  end;
  {...}
class function TgeomForm.dimension :
Integer;
begin
  result := 2;
end;
```

◆ Exceptions - Ausnahmebehandlung

- wird gefeuert wenn ein unerwarteter Fehler auftritt
 - z. B.: wenn eine Variable null ist

```
try
    c := a / b;
except
    exit;
end;
```

```
try
    c := a / b;
finally
    myObject.free;
end;
```

◆ Besonderes

Hinweis: Ist eine Prozedur nicht im Interface-Teil deklariert, so kann man sie nur an einer Stelle aufrufen, welche unterhalb dieser Prozedur liegt.

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
    foo;  
end;  
procedure foo;  
begin  
    ShowMessage('foo');  
end;
```

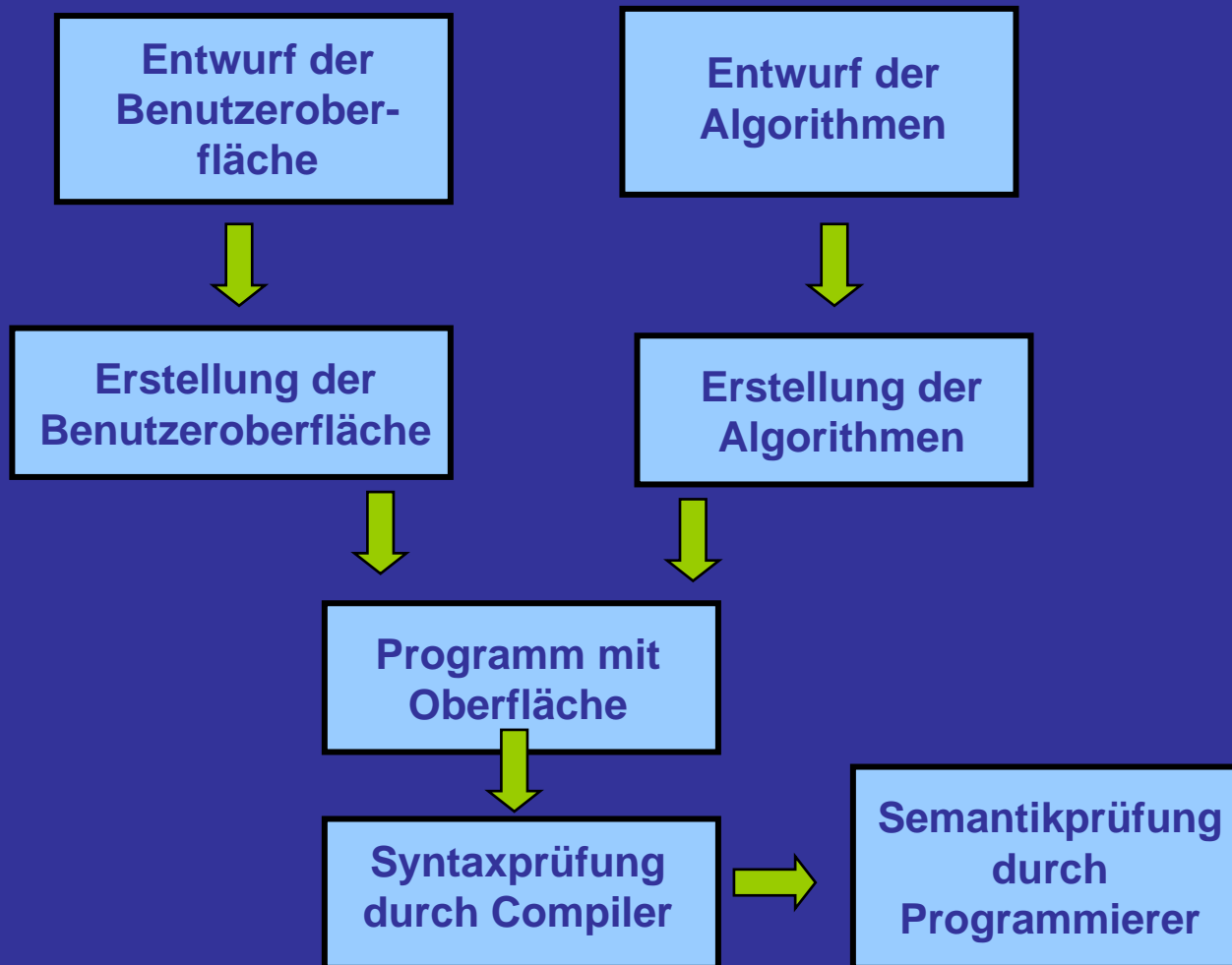
◆ Konstruktor – Anlegen eines Objektes

```
private
Fhoehe : Integer;
Fbreite : Integer;
public
constructor create(hoehe, breite : Integer); overload;
constructor create(groesse : Integer); overload;
end;

constructor TRechteck.create(hoehe, breite: Integer);
begin
inherited create;
FHoehe := hoehe;
FBreite := breite;
end;

constructor TRechteck.create(groesse : Integer);
begin
inherited create;
FHoehe := groesse;
FBreite := groesse;
end;
```

◆ Entwurf eines graphischen Programmes



◆ Infos über die Entwicklungsumgebung

- Spin-off von Borland IDEs in Codegear → neue alte Turbo-Reihe
 - kostenlose **Explorer-Version**
 - frei, nur die 200 mitgelieferter Standardkomponenten nutzbar
 - keine Einbindung externer Komponenten von Drittherstellern
 - Professional Version
 - 400 Euro
 - nur eine Turbo-Version gleichzeitig installierbar (technische Hintergründe, keine lizenzrechtlichen)
- Turbo-Versionen sind abgespeckte Borland Developer Studios 2006

Fragen?



Antworten!

Vielen Dank für ihre Aufmerksamkeit!

19.12.2005

„Einführung in die Programmierung“

Tobias Giese

Unterlagen:

<http://www.tobias-giese.de>

“An investment in knowledge pays the best return.”
- Benjamin Franklin -