

C - Kurs (1)

Einführung, Kontrollstrukturen, Debugger

Wolfgang Hönic

24. Oktober 2009

Überblick

- 1 Hello World
 - Einführung
 - Visual Studio (2008)
- 2 Variablen
 - Deklaration
 - Ein- und Ausgabe
 - Datenverarbeitung
- 3 Kontrollstrukturen
 - Bedingungen
 - If-Anweisung
 - Schleifen
- 4 Debuggen
 - Einführung
 - Visual Studio (2008)
- 5 Achilles

Motivation

Warum gerade C?

- C genießt bei systemnaher Programmierung immer noch große Verbreitung
 - Linux-Kernel
 - eingebette Systeme (Auto, MP3-Player, Waschmaschine)
- schwierige Konzepte in C (z.B. Zeiger) helfen beim Verständnis was auf einem Rechner abläuft
- Grundlage für neuere Sprachen: C++, C#, Java, PHP
- Algorithmen und Datenstrukturen

Hello World

hello.c

```
"Hello World!"
```

- `printf(...)`: Funktion für Bildschirmausgabe
- Jede Anweisung wird mit Semikolon (;) abgeschlossen
- `main`: Startpunkt ins Programm
- `include`: Einbinden eines vorhandenen Moduls
- `stdio`: Funktionen für "standard in- and output"

Hello World

hello.c

```
printf("Hello World!");
```

- `printf(...)`: Funktion für Bildschirmausgabe
- Jede Anweisung wird mit Semikolon (;) abgeschlossen
- `main`: Startpunkt ins Programm
- `include`: Einbinden eines vorhandenen Moduls
- `stdio`: Funktionen für "standard in- and output"

Hello World

hello.c

```
printf("Hello World!");
```

- printf(...): Funktion für Bildschirmausgabe
- Jede Anweisung wird mit Semikolon (;) abgeschlossen
- main: Startpunkt ins Programm
- include: Einbinden eines vorhandenen Moduls
- stdio: Funktionen für "standard in- and output"

Hello World

hello.c

```
int main() {  
    printf("Hello World!");  
    return 0;  
}
```

- printf(...): Funktion für Bildschirmausgabe
- Jede Anweisung wird mit Semikolon (;) abgeschlossen
- main: Startpunkt ins Programm
- include: Einbinden eines vorhandenen Moduls
- stdio: Funktionen für "standard in- and output"

Hello World

hello.c

```
#include <stdio.h>

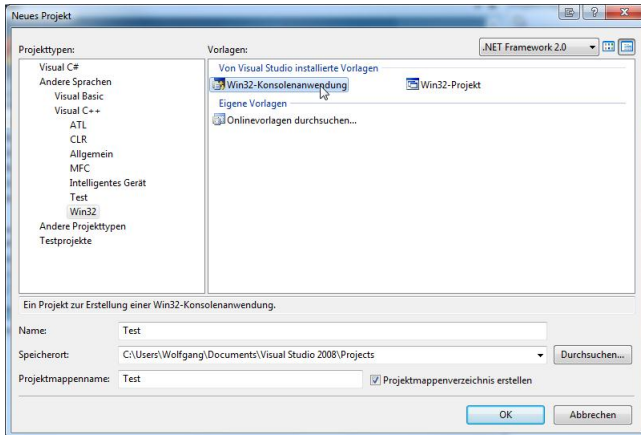
int main() {
    printf("Hello World!");
    return 0;
}
```

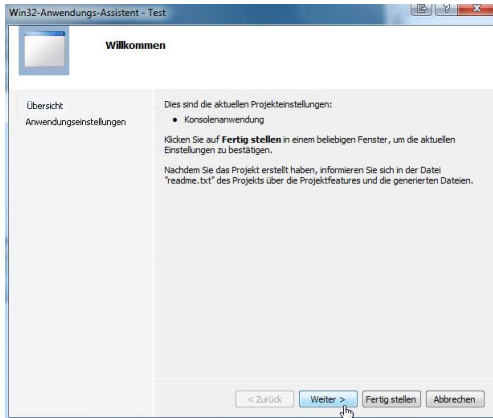
- `printf(...)`: Funktion für Bildschirmausgabe
- Jede Anweisung wird mit Semikolon (;) abgeschlossen
- `main`: Startpunkt ins Programm
- `include`: Einbinden eines vorhandenen Moduls
- `stdio`: Funktionen für "standard in- and output"

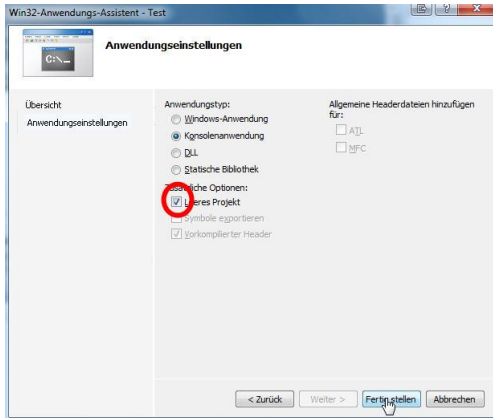
Hello World

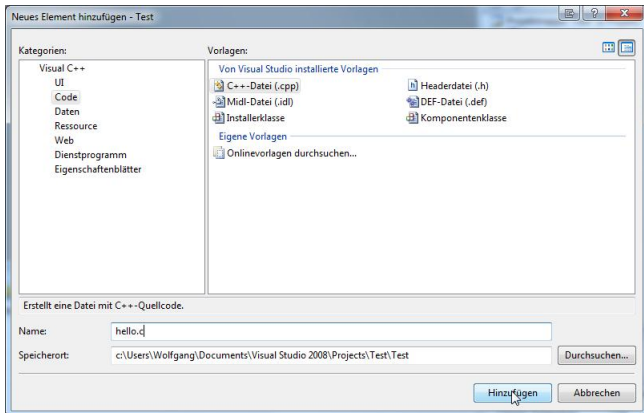
Aufgabe 1

Programmieren Sie zum Kennenlernen der Entwicklungsumgebung ein Hello-World-Programm und testen Sie dieses.









The screenshot shows the Microsoft Visual Studio 2008 IDE. The main window displays the source code for a file named 'hello.c'. The code is as follows:

```

(Global Gültigkeitsbereich)
main()
#include <stdio.h>

int main(void) {
    printf("Hello World!");

    return 0;
}
  
```

The Project Explorer on the right shows a project named 'Test' containing a sub-project 'Test (1 Projekt)' with folders for 'Headerdateien', 'Quelldateien', and 'Ressourcendateien'. The 'Quelldateien' folder contains the file 'hello.c'.

The Error List window at the bottom shows 0 errors, 0 warnings, and 0 messages. The Properties window at the bottom right shows the properties for the 'main' function, including its name and file path.

The status bar at the bottom indicates 'Gespeicherte(s) Element(e)', 'Z 7', 'S 2', 'Zei 2', and 'EINFG'.

The screenshot shows the Visual Studio 2008 interface. The main window displays a C program named 'hello.c' with the following code:

```

(Globaler Gültigkeitsbereich)
#include <stdio.h>

int main(void) {
    printf("Hello World!");

    return 0;
}
    
```

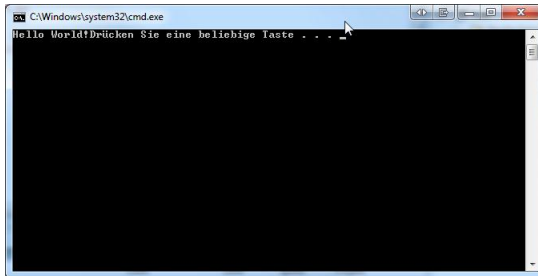
The 'Debuggen' menu is open, showing the following options:

- Fenster
- Debugging starten (F5)
- Starten ohne Debugging (Strg+F5)
- An den Prozess anhängen...
- Ausnahmen... (Strg+D, E)
- Einzelschritt (F11)
- Prozedurschritt (F10)
- Haltepunkt umschalten (F9)
- Neuer Haltepunkt
- Alle Haltepunkte löschen (Strg+Umschalt+F9)

The 'Fehlerliste' (Error List) window at the bottom shows 0 errors, 0 warnings, and 0 messages. The 'Projektmappe-Explorer' (Solution Explorer) shows a project named 'Test' with files: 'Headerdateien', 'Quelldateien', 'hello.c', and 'Ressourcendateien'. The 'Eigenschaften' (Properties) window shows the 'main' function with the following details:

Eigenschaften	
main VCCodeFunction	
(Name)	main
File	c:/users/wolfgang/docum...
FullName	main
(Name)	
Legt den Namen des Objekts fest oder gibt ihn zurück.	

The status bar at the bottom indicates 'Gespeicherte(s) Element(e)', 'Z 7', 'S 2', 'Zei 2', and 'EINFG'.



Variablen: Deklaration (1)

- **Variablen** sind Gedächtnisstützen
- \Rightarrow benötigen Namen und Typ (int, float, char, ...)
 - beides muss dem Rechner vor der Nutzung bekannt gegeben werden: **Deklaration**
 - Rechner ersetzt Namen zur Laufzeit mit dem zugehörigen Wert
- können gelesen und verändert (**zugewiesen**) werden

Deklaration

```
datentyp name_1 [= wert_1], ..., name_n [=wert_n];
```

Deklaration (Beispiel)

```
int weg;  
int geschwindigkeit = 0, zeit = 1;
```

Variablen: Deklaration (2)

Variablenamen

- möglichst aussagekräftig (nur einzelne Buchstaben ungünstig)
- Bestehen **genau** aus den Zeichenbereichen a-z, A-Z, 0-9, _
- eine Ziffer (0-9) darf nicht erstes Zeichen sein
- keine Sonderzeichen (z.B. ? , . ; ä ü ö ß) enthalten
- dürfen nicht wie Schlüsselwörter (z.B. if, while, for) lauten
- Unterscheidung zwischen Groß- und Kleinschreibung

Variablen: Deklaration (3)

Datentypen

```
char c = '?'; /* character: ein einzelnes  
Zeichen */  
int i = 23; /* integer: ganze Zahlen.  
Wertebereich meist von  $-2^{31}$  bis  $2^{31} - 1$   
*/  
float f = 1.2E6; // floating point number:  
Gleitkommazahl; Punkt statt Komma als  
Dezimaltrennzeichen
```

Ausgabe

Ausgabe (allgemein)

```
printf("%format1 ... %format2 ...", v1, v2);
```

Mehr Infos zu den Formatstrings: [http://man.cx/printf\(3\)/de](http://man.cx/printf(3)/de)

Ausgabe (Beispiele)

```
printf("Hello World");  
printf("%i", integervariable);  
printf("Geschwindigkeit: %f", speed);  
printf("float: %f char: %c int: %i", f,c,i);  
printf("neue Zeile:\n");
```

Eingabe

Eingabe (allgemein)

```
scanf("%format", &variable);
```

Mehr Infos zu den Formatstrings: [http://man.cx/scanf\(3\)/de](http://man.cx/scanf(3)/de)

Eingabe (Beispiele)

```
scanf("%i", &integervariable);  
scanf("%f", &floatvariable);  
scanf("%c", &charvariable);
```

Datenverarbeitung

Rechenoperationen

- + Addition
- Substraktion
- * Multiplikation
- / Division (wird bei Integer abgerundet)
- % Modulo (Rest bei ganzzahliger Division)
- Punkt- vor Strichrechnung
- Variable für Zuweisung steht links (**left-hand-side-value**)

Variablen: Beispiel

vars.c

```
#include <stdio.h> /*für printf*/
int main() {
    int v, s, t; /* Deklaration */
    s = 10; /* Zuweisung [Meter] */
    t = 5; /* Zuweisung [Sekunden] */
    v = s / t; /* Berechnung [m/s] */
    printf("Geschwindigkeit: %i", v);
    return 0;
}
```

- /* ist ein Kommentar; Alles was dazwischen steht, wird vom Compiler ignoriert */

Variablen: Aufgaben

Aufgabe 2

- 1 Erweitern Sie das Hello-World-Programm zur Berechnung einer Geschwindigkeit.
- 2 Der Weg s und die Zeit t sollen nun zur Laufzeit vom Nutzer eingegeben werden können. Nutzen Sie die Funktion `scanf` um einen Eingabewert von der Konsole zu lesen.
- 3 Welche Probleme ergeben sich bei ungünstigen Eingabewerten? (Z.B. $s = 5$ und $t = 3$).

Bedingungen

- (wert1 vergleichsoperator wert2)

Vergleichsoperatoren

wert1	==	wert2	wert1 ist gleich	wert2
wert1	>	wert2	wert1 ist größer als	wert2
wert1	<	wert2	wert1 ist kleiner als	wert2
wert1	>=	wert2	wert1 ist größer gleich als	wert2
wert1	<=	wert2	wert1 ist kleiner gleich als	wert2
wert1	!=	wert2	wert1 ist ungleich	wert2

logische Verknüpfung

bedingung1	& &	bedingung2	UND
bedingung1		bedingung2	ODER

Bedingungen

Beispiele

```
(x<5) /* x kleiner als 5 */  
(x!=y) /* x ungleich y */  
(1!=2) /* immer wahr */  
((x<5)&&(x>3)) /* x zwischen 3 und 5 */  
((y>5)|| (x<3)) /* y größer 5 oder x kleiner 3  
    (oder beides) */
```

If-Anweisung

Allgemein

```
if (Bedingung)
    TuWas; /* falls Bedingung erfüllt */
else /* optional */
    TuWasAnderes; /* falls Bedingung nicht
        erfüllt */
```

If-Anweisung

Allgemein - mehrere Befehle

```
if (Bedingung)
{
    Befehl1;
    ...
    Befehl99;
}
else
{
    Befehl;
}
```

If-Anweisung

Allgemein - mehrere Befehle, ohne else

```
if (Bedingung)
{
    Befehl1;
    ...
    Befehl99;
}
```

If-Anweisung

Beispiel

```
int x;  
scanf("%i",&x);  
if ((x%2) == 0)  
{  
    printf("x ist durch 2 teilbar!");  
}  
else  
    printf("x ist ungerade!");
```

switch-case

Beispiel

```
int note;  
switch(note) {  
    case 1: printf("sehr gut"); break;  
    case 2: printf("gut"); break;  
    case 3: printf("befriedigend"); break;  
    case 4: printf("ausreichend"); break;  
    case 5: printf("mangelhaft"); break;  
    case 6: printf("ungenügend"); break;  
    default: printf("seltsame Bewertung");  
        break;  
}
```

switch-case: Vergleich mit If

Beispiel

```
int note;  
if(note == 1) { printf("sehr gut"); }  
else if(note == 2) { printf("gut"); }  
else if(note == 3) { printf("befriedigend"); }  
else if(note == 4) { printf("ausreichend"); }  
else if(note == 5) { printf("mangelhaft"); }  
else if(note == 6) { printf("ungenügend"); }  
else { printf("seltsame Bewertung"); }
```

Schleifen

- zur Wiederholung von Befehlen
- drei verschiedene Arten:
 - while: prüft am Anfang und nach jedem Schleifendurchlauf eine Bedingung
 - do-while: prüft nur nach jedem Schleifendurchlauf die Bedingung \Rightarrow wird mindestens einmal durchlaufen
 - for: kombiniert while-schleife mit Zählvariable

Schleifen

while

```
int x = 1;
while(x<=10) {
    printf("x = %i", x);
    x = x + 1;
}
```

for

```
int x;
for(x=1;x<=10;x=x+1) {
    printf("x = %i", x);
}
```

do-while

```
int x = 1;
do {
    printf("x = %i", x);
    x = x + 1;
}
while(x<=10);
```

Aufgabe

Aufgabe 3

Programmieren Sie ein C-Programm welches von der Kommandozeile eine natürliche Zahl n einliest und die Summe der ersten n natürlichen Zahlen berechnet und ausgibt:

$$\text{Ausgabe} = \sum_{i=1}^n i$$

Nutzen Sie zur Fehlerbehandlung der Eingabe (negative Zahlen) eine passende Kontrollstruktur.

Debuggen

- Debuggen (Entwanzen) dient der Fehlersuche
- Quelltext kann schrittweise ausgeführt werden
- Variablen können dabei angezeigt und verändert werden
- **Breakpoint:** Stelle an der angehalten werden soll
- Auch für andere Dinge nützlich

Debuggen

debug.c

```
#include <stdio.h>

int main() {
    int x = 1024;
    int sh = 2;
    int i;
    /* i++ bedeutet: i = i + 1 */
    for(i=0;i<10;i++)
        x = x >> sh;
    return 0;
}
```

Debuggen

Aufgabe 4

Finden Sie durch Nutzung des Debuggers heraus, was der `>>` Operator bewirkt. Untersuchen Sie dazu die Zeile `x = x >> sh` mit verschiedenen Werten von `x` und `sh`.

The screenshot shows the Visual Studio 2008 IDE with the following components:

- Menu Bar:** Datei, Bearbeiten, Ansicht, Projekt, Erstellen, Debuggen, Extras, Test, Fenster, Hilfe.
- Toolbar:** Standard Visual Studio tool icons.
- Code Editor:** Displays the source code for 'hello.c' with the following content:


```

            (Globaler Gültigkeitsbereich)
            main()
            #include <stdio.h>

            int main(void) {
                int x = 1024;
                int sh = 2;
                int i;

                // i++ bedeutet: i = i + 1
                for(i=0; i<10; i++)
                    x = x >> sh;

                return 0;
            }
            
```
- Solution Explorer:** Shows a project named 'Test' containing folders for 'Headerdateien', 'Quelldateien', and 'Ressourcendateien', with a file named 'hello.c' under 'Quelldateien'.
- Eigenschaften (Properties) Window:** Shows properties for the 'main' function, including Name, File, and FullName.
- Fehlerliste (Error List) Window:** Shows 0 Fehler, 0 Warnungen, and 0 Meldungen. The table below represents its structure:

Beschreibung	Datei	Zeile	Spalte	Projekt

At the bottom of the IDE, the status bar shows 'Bereit', 'Z10', 'S1', 'Zei 1', and 'EINFG'.

Test - Microsoft Visual Studio

Datei Bearbeiten Ansicht Projekt Erstellen Debuggen Extras Test Fenster Hilfe

Debug Win32

hello.c

```
(Globaler Gültigkeitsbereich)
main()
#include <stdio.h>

int main(void) {
    int x = 1024;
    int sh = 2;
    int i;

    // i++ bedeutet: i = i + 1
    for(i=0; i<10; i++)
        x = x >> sh;

    return 0;
}
```

Projektmappen-Explorer - Projektmappe "Test" ...

- Projektmappe "Test" (1 Projekt)
- Test
 - Headerdateien
 - Quelldateien
 - hello.c
 - Ressourcendateien

Fehlerliste

0 Fehler 0 Warnungen 0 Meldungen

Beschreibung	Datei	Zeile	Spalte	Projekt
--------------	-------	-------	--------	---------

Fehlerliste Suchergebnisse: 1 Ergebnisse der Symbolsuche

Eigenschaften

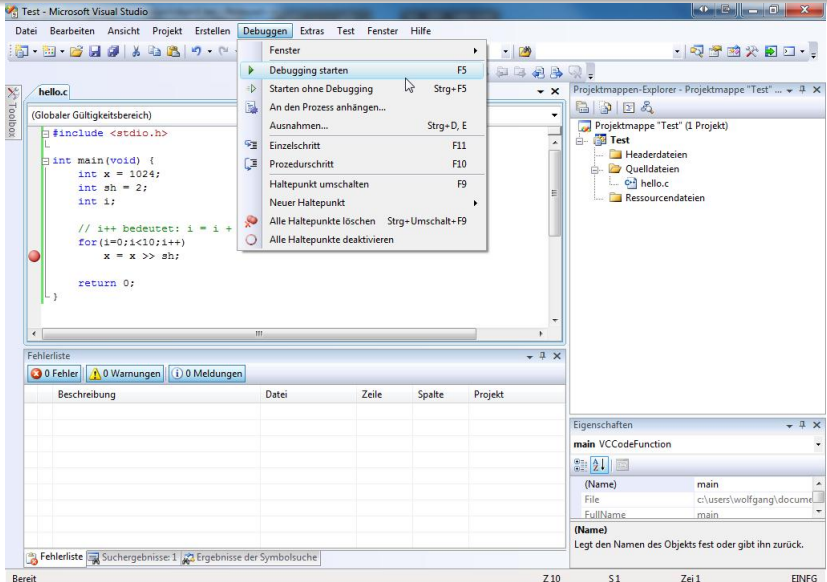
main VCCodeFunction

(Name)	main
File	c:/users/wolfgang/documc
FullName	main

(Name)
Legt den Namen des Objekts fest oder gibt ihn zurück.

Bereit Z10 S21 Zei15 EINF

Breakpoint: Erstellen durch Doppelklicken am Rand



Test (Debugging) - Microsoft Visual Studio

Datei Bearbeiten Ansicht Projekt Erstellen Debuggen Extras Test Fenster Hilfe

Debug Win32

hello.c

(Globaler Gültigkeitsbereich) main()

```
#include <stdio.h>

int main(void) {
    int x = 1024;
    int sh = 2;
    int i;

    // i++ bedeutet: i = i + 1
    for(i=0;i<10;i++)
        x = x >> sh;

    return 0;
}
```

Führt Einzelschritt aus (F10)

Lokal

Name	Wert	Typ
sh	2	int
i	0	int
x	1024	int

Aufrufliste

Name	Sprache
Test.exe!main() Zeile 10	C
Test.exe!_tmainCRTStartup() Zeile 586 + 0x19 Bytes	C
Test.exe!mainCRTStartup() Zeile 403	C
kernel32.dll!76b93677()	
[Unten angegebene Rahmen sind möglicherweise nicht korrekt und/oder fehlen, keine Symbole]	
ntdll.dll!77e79d72()	
ntdll.dll!77e79d45()	

Zeigt aktuelle Werte an. Diese sind auch veränderbar

Fehlerliste Lokal Überwachen 1

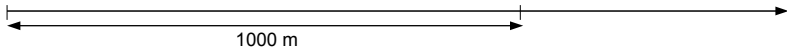
Bereit Z10 S1 Zei1 EINFG

Achilles und die Schildkröte

- Achilles ist 10x so schnell wie die Schildkröte.
- Die Schildkröte hat 1000m Vorsprung.
- Wenn Achilles die 1000m gerannt ist, ist die Schildkröte schon 100m weiter.
- Nach den 100m ist die Schildkröte 10m weiter.
- Schafft es Achilles die Schildkröte zu überholen?

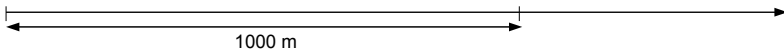
Achilles und die Schildkröte

$$t_0 = 0 \quad s_{\text{Achilles}0} = 0 \quad s_{\text{Schildkroete}0} = 1000$$

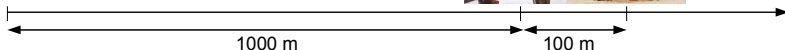


Achilles und die Schildkröte

$$t_0 = 0 \quad s_{\text{Achilles}0} = 0 \quad s_{\text{Schildkroete}0} = 1000$$



$$t_1 = \frac{1000}{v_{\text{Achilles}}} \quad s_{\text{Achilles}1} = 1000 \quad s_{\text{Schildkroete}1} = 1000 + v_{\text{Schildkroete}} t_1$$



Achilles und die Schildkröte

$$t_0 = 0 \quad s_{Achilles0} = 0 \quad s_{Schildkroete0} = 1000$$

$$t_1 = \frac{1000}{v_{Achilles}} \quad s_{Achilles1} = 1000 \quad s_{Schildkroete1} = 1000 + v_{Schildkroete} t_1$$

$$t_2 = \frac{v_{Schildkroete} t_1}{v_{Achilles}} \quad s_{Achilles2} = 1100 \quad s_{Schildkroete2} = 1100 + v_{Schildkroete} t_2$$

$$t_3 = \frac{v_{Schildkroete} t_2}{v_{Achilles}} \quad s_{Achilles3} = 1110 \quad s_{Schildkroete3} = 1110 + v_{Schildkroete} t_3$$

Achilles und die Schildkröte

$$t_0 = 0 \quad s_{Achilles0} = 0 \quad s_{Schildkroete0} = 1000$$

$$t_1 = \frac{1000}{v_{Achilles}} \quad s_{Achilles1} = 1000 \quad s_{Schildkroete1} = 1000 + v_{Schildkroete} t_1$$

$$t_2 = \frac{v_{Schildkroete} t_1}{v_{Achilles}} \quad s_{Achilles2} = 1100 \quad s_{Schildkroete2} = 1100 + v_{Schildkroete} t_2$$

$$t_3 = \frac{v_{Schildkroete} t_2}{v_{Achilles}} \quad s_{Achilles3} = 1110 \quad s_{Schildkroete3} = 1110 + v_{Schildkroete} t_3$$

Achilles und die Schildkröte

$$t_0 = 0 \quad s_{Achilles0} = 0 \quad s_{Schildkroete0} = 1000$$

$$t_1 = \frac{1000}{v_{Achilles}} \quad s_{Achilles1} = 1000 \quad s_{Schildkroete1} = 1000 + v_{Schildkroete} t_1$$

$$t_2 = \frac{v_{Schildkroete} t_1}{v_{Achilles}} \quad s_{Achilles2} = 1100 \quad s_{Schildkroete2} = 1100 + v_{Schildkroete} t_2$$

$$t_3 = \frac{v_{Schildkroete} t_2}{v_{Achilles}} \quad s_{Achilles3} = 1110 \quad s_{Schildkroete3} = 1110 + v_{Schildkroete} t_3$$

Schildkrötenmathematik

Zeit die Achilles für eine Etappe benötigt:

$$t_n = \frac{S_{\text{Schildkroete}}}{v_{\text{Achilles}}} = \frac{t_{n-1} \cdot v_{\text{Schildkroete}}}{v_{\text{Achilles}}}$$

Erste Etappe:

$$t_1 = \frac{S_{\text{Vorsprung}}}{v_{\text{Achilles}}}$$

Es gibt aber unendlich viele Etappen:

$$t_{\text{ueberholen}} = \sum_{n=1}^{\infty} t_n$$

Wird Achilles gewinnen?

Aufgabe 5

- 1 Programmieren Sie ein C-Programm welches von der Kommandozeile alle nötigen Parameter für das Schildkrötenwettrennen einliest und die Zeit $t_{ueberholen}$ näherungsweise berechnet. Da eine Summation bis unendlich nicht möglich ist, geben Sie alle t_n auf der Konsole von $1 \leq n \leq 20$ aus.
- 2 Die Obergrenze 20 wurde empirisch ermittelt. Welches Abbruchkriterium könnte dies automatisieren?
- 3 Ermitteln Sie auf anderem Wege eine Lösung für das Problem und vergleichen Sie dieses mit der Lösung ihres C-Programms.

Blick über den Tellerrand: Schildkrötenphysik

$s(t)$ für Achilles:

$$s_{Achilles}(t) = v_{Achilles} \cdot t$$

$s(t)$ für Schildkröte:

$$s_{Schildkroete}(t) = v_{Schildkroete} \cdot t + s_{Vorsprung}$$

Gleichsetzen und nach t umstellen ergibt:

$$t_{ueberholen} = \frac{s_{Vorsprung}}{v_{Achilles} - v_{Schildkroete}}$$

Blick über den Tellerrand: Schildkrötengrenzwertrechnung

$$t_1 = \frac{SVorsprung}{v_{Achilles}} \quad q = \frac{v_{Schildkroete}}{v_{Achilles}}$$

$$t_{ueberholen} = \sum_{n=1}^{\infty} t_n = t_1 + t_1 \frac{v_{Schildkroete}}{v_{Achilles}} + t_1 \left(\frac{v_{Schildkroete}}{v_{Achilles}} \right)^2 + \dots$$

Nutzung der geometrischen Reihe:

$$\begin{aligned} t_{ueberholen} &= t_1 \sum_{n=0}^{\infty} q^n = t_1 \frac{1}{1-q} = \frac{SVorsprung}{v_{Achilles}} \frac{1}{1 - \frac{v_{Schildkroete}}{v_{Achilles}}} \\ &= \frac{SVorsprung}{v_{Achilles}} \frac{1}{\frac{v_{Achilles} - v_{Schildkroete}}{v_{Achilles}}} = \frac{SVorsprung}{v_{Achilles} - v_{Schildkroete}} \end{aligned}$$

Achilles und die Schildkröte

Sinn und Zweck der letzten beiden Folien

Programmieren ist auch für das (näherungsweise) Lösen schwieriger mathematischer und physikalischer Probleme nutzbar!

zum Schluss...

vielen Dank für die
Aufmerksamkeit

Feedback erwünscht!

<http://ckurs.rextech.de>