# Einführung in die Programmierung

## Brückenkurs

Prof. Dr. Rethmann

Fachbereich Elektrotechnik und Informatik
Hochschule Niederrhein

WS 2009/10

# Übersicht

*Montag, 14. September 2009*

- *Ein kleines Programm*
- Eingangstest Informatik
- Was ist Informatik?

# Ein erstes kleines Programm

*Hello, World!*

```
#include <stdio.h>

void main(void) {
    printf("Hello,\nWorld!\n");
}
```

*Erklärungen:*

- Mittels `#include <stdio.h>` wird eine Bibliothek bereitgestellt, die Funktionen zur Ein- und Ausgabe enthält.
- Der Start eines Programms besteht im Ausführen der Funktion `main`.
- Alle Anweisungen werden mit einem Semikolon beendet.

# Ein erstes kleines Programm

*Erklärungen:* (Fortsetzung)

- Die Funktion `printf()` gibt eine Zeichenkette auf dem Bildschirm aus.
  Solche Standardfunktionen sind übersetzte Funktionen, die zur C-Implementierung gehören.
- eine Zeichenkette ist durch doppelte Anführungsstriche am Anfang und Ende gekennzeichnet, z.B. `"James Bond"`
- Anweisungsfolgen werden mit geschweiften Klammern `{` und `}` zusammengefasst, der geklammerte Block gilt als eine Anweisung.
- das Zeichen `\n` bedeutet *new line*, es bewirkt also einen Zeilenvorschub

# Ein zweites kleines Programm

*Variablen:*

```
#include <stdio.h>
void main(void) {
    int i = 5;
    printf("i = %d\n", i);
    i = i + 2;
    printf("i = %d\n", i);
}
```

*Erklärungen:*

- alle Variablen in C haben einen Typ, im Beispiel definieren wir `i` vom Typ ganze Zahl
- Variablen können in Ausdrücken wie `i = i + 2` verwendet werden: erhöhe den Wert von `i` um zwei und weise diesen neuen Wert der Variablen `i` zu

# Ein zweites kleines Programm

*Hinweis:*

- Variablen in C nehmen zu unterschiedlichen Zeiten unterschiedliche Werte an

```
int i = 1;
while (i < 10) {
    printf("sqr(%d) = %d\n", i, i * i);
    i = i + 1;
}
```

- Variablen in der Mathematik sind Platzhalter für feste Werte

$$\begin{aligned} 3x + 4y &= 20 \\ 2x + 17y &= 42 \end{aligned}$$

also $x = 4$, $y = 2$

# Ein drittes kleines Programm

*Schleifen:*

```
#include <stdio.h>
void main(void) {
    int i = 5;
    while (i < 10) {
        printf("i = %d\n", i);
        i = i + 2;
    }
}
```

*Erklärungen:*

- solange der Wert von `i` kleiner ist als 10, wird der Rumpf der Schleife ausgeführt
- sobald der Wert von `i` gleich 10 oder größer ist, wird die Abarbeitung des Schleifenrumpfs abgebrochen (evtl. schon zu Beginn)

# Ein viertes kleines Programm

*Zählschleifen:*

```
#include <stdio.h>
void main(void) {
    int i;
    for (i = 1; i < 10; i += 1) {
        printf("i = %d\n", i);
    }
}
```

*Erklärungen:*

- der erste Ausdruck ist der *Initialisierungsausdruck*, der vor Beginn der Schleife einmal ausgeführt wird
- solange der zweite Ausdruck erfüllt ist, wird der Schleifenrumpf durchlaufen
- nach jedem Schleifendurchlauf wird der dritte Ausdruck bewertet $\longrightarrow$ Schleifenvariablen ändern

# Ein fünftes kleines Programm

*Leerzeilen:*

```
#include <stdio.h>

void main(void) {
    int i = 5;

    while (i < 10) {
        printf("i = %d\n", i);
        i = i + 2;
    }
}
```

*Erklärungen:* Leerzeilen haben keine syntaktische Bedeutung, aber sie erhöhen die Lesbarkeit des Programms

- vor Funktionen
- nach Deklaration von Variablen

# Ein sechstes kleines Programm

```
#include <stdio.h>

void main(void) {
    int i, anf, end;

    printf("Anfangswert? ");
    scanf("%d", &anf);
    printf("Endwert? ");
    scanf("%d", &end);

    i = anf;
    while (i < end) {
        printf("i = %d\n", i);
        i = i + 1;
    }
}
```

# Ein sechstes kleines Programm

*Erklärung:* die Funktion `scanf()` liest Werte von der Tastatur ein

- dazu muss zum einen der erwartete Datentyp angegeben werden

| | |
|---|---|
| `%d` | `int` |
| `%f` | `float` |
| `%c` | `char` |

- zum anderen das Ziel, also die Variable, in der der eingelesene Wert gespeichert werden soll

# Ein siebtes kleines Programm

*Verzweigungen:*

```
#include <stdio.h>

void main(void) {
     int i;

     printf("Wert? ");
     scanf("%d", &i);

     if (i % 2 == 0)
          printf("%d ist gerade\n", i);
     else printf("%d ist ungerade\n", i);
}
```

# Ein siebtes kleines Programm

*Erklärungen:*

- Mittels Auswahlanweisungen kann der Ablauf eines Programms abhängig von Bedingungen geändert werden.
- Der Modulo-Operator `%` bestimmt den ganzzahligen Rest bei einer Division.
- Im Beispiel wird der Ausdruck `i % 2 == 0` bewertet. Falls er wahr ist, wird die darauf folgende Anweisung ausgeführt, ansonsten die Anweisung im `else`-Zweig
- Der `else`-Zweig kann entfallen

# Übersicht

*Montag, 14. September 2009*

- Ein kleines Programm
- *Eingangstest Informatik*
- Was ist Informatik?

# Einstufungstest

Addiere folgende Zahlen im Binärsystem

$$\begin{array}{rr} & 101001 \\ + & 010011 \end{array}$$

Die Summe beträgt

- ☐ 110110
- ☒ 111100
- ☐ 110011
- ☐ 111111

# Einstufungstest

Die Booleschen Operatoren $\wedge$ (AND) und $\vee$ (OR) sind folgendermaßen definiert:

| $\wedge$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| $\vee$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Wann ist der folgende Ausdruck wahr, d.h. 1?
(A ODER B) UND (C ODER D) bzw. $(A \vee B) \wedge (C \vee D)$

- ☒ A=1, B=0, C=0, D=1
- ☐ A=0, B=1, C=0, D=0
- ☒ A=0, B=1, C=1, D=1
- ☐ A=0, B=0, C=1, D=1
- ☒ A=1, B=0, C=1, D=1

# Einstufungstest

Was ist eine gültige Internet Adresse laut IPv4?

- ☐ 104.275.94.1
- ☒ 194.94.121.248
- ☐ 111.945.76
- ☐ 54.236.51.12.66

# Einstufungstest

Was ist ein Compiler?

- ☐ Ein Programm zur Verschlüsselung.
- ☐ Ein Hardware-Bauteil.
- ☒ Ein Programm zur Übersetzung von Programmcode.
- ☐ Ein Computervirus.

# Einstufungstest

Gegeben sei folgender Algorithmus:

```
a := 1;
b := 0;
Solange a < 3 tue {
    wenn a gerade, dann b := b + 1;
    a := a + 1;
}
gib b zurück;
```

Welchen Wert haben die Variablen a und b nach jedem Durchlauf der Schleife?

| Durchlauf | Wert von a | Wert von b |
|---|---|---|
| 1 | 2 | 0 |
| 2 | 3 | 1 |

# Einstufungstest

Die folgende rekursive Berechnungsvorschrift liefert den größten gemeinsamen Teiler zweier positiver ganzer Zahlen:

$$ggT(x,y) = \begin{cases} x & \text{falls} \quad x = y \\ ggT(x-y,y) & \text{falls} \quad x > y \\ ggT(x,y-x) & \text{falls} \quad x < y \end{cases}$$

Berechnen Sie $ggT(16,6)$ durch aufeinanderfolgendes Anwenden der Rekursionsvorschrift.

$$\begin{aligned} ggT(16,6) &= ggT(10,6) \\ &= ggT(4,6) \\ &= ggT(4,2) \\ &= ggT(2,2) \\ &= 2 \end{aligned}$$

# Einstufungstest

C-Programmierung: Was ist der Wert der Variablen `x` am Ende der Programmzeilen?

```
int i;
int x = 0;

for (i = 1; i < 5; i = i + 1)
    x = x + i;
```

- ☐ 4
- ☐ 5
- ☐ 8
- ☒ 10

# Einstufungstest

C-Programmierung: Was ist der Wert der Variablen `x` am Ende der Programmzeilen?

```
int i = 5;
int x = 0;

while (i < 8) {
    if (i == 6)
        x = x - i;
    else
        x = x + i;
    i = i + 1;
}
```

- ☐ 4
- ☒ 6
- ☐ 10
- ☐ 18

# Einstufungstest

C-Programmierung: Welche Ausgabe erzeugt folgendes Programm?

```c
#include <stdio.h>
int main(void) {
    int a[] = {1, 2, 3, 4};
    int sum[5];
    int i = 0;

    sum[0] = 0;
    while (i < 4) {
        sum[i+1] = sum[i] + a[i];
        i = i + 1;
    }
    for (i = 0; i <= 4; i++)
        printf("%3d\n", sum[i]);
}
```

- ☐ 0, 1, 2, 3, 4
- ☒ 0, 1, 3, 6, 10
- ☐ 1, 2, 4, 7, 10
- ☐ 1, 3, 6, 10, 15

# Einstufungstest

C-Programmierung: Welche Ausgabe erzeugt folgendes Programm?

```
#include <stdio.h>

int fkt(int n) {
    if (n == 0)
        return 1;
    return n * fkt(n-1);
}

int main(void) {
    printf("%d\n", fkt(4));

    return 0;
}
```

- ☐ 1
- ☐ 10
- ☒ 24
- ☐ 36

# Einstufungstest

Schreiben Sie einen logischen Ausdruck in C-Notation für folgende Tabelle:

| A | B | C | erg |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

- ☐ `(A = B = C)`
- ☐ `(A && B && C)`
- ☒ `!C`
- ☒ `C == 0`

# Übersicht

*Montag, 14. September 2009*

- Ein kleines Programm
- Eingangstest Informatik
- *Was ist Informatik?*

# Informatik

## *Was ist das?*

- Kunstwort aus *Infor*mation und Mathe*matik*
- Informatik ist eng mit Computern verknüpft: solange es keine Computer gab, gab es auch keine Informatik
- elektronische Rechenmaschine entstand um 1940

## *Ursprung*

- Rechnen galt bis Anfang der Neuzeit als *Kunst*
- heute kann jeder die vier Grundrechenarten ausführen
  - mechanisch ausführbares Verfahren, dass nicht verstanden werden muss, um es anwenden zu können

  $\rightarrow$ kann einer Maschine übertragen werden kann

*Algorithmus*

- mechanisch ausführbares Rechenverfahren
- bildet den Kern der Informatik
- benannt nach dem persischen Mathematiker Abu Ja'far Mohammed ibn Mûsâ al-Khowârizmî

*Beispiel:* einfacher Primzahltest

*eine Zahl* $p \in \mathbb{N}$ *ist eine Primzahl*
$\iff$ *keine Zahl* $z \in \{2, 3, \ldots, p-1\}$ *teilt* $p$

*Algorithmus* Primzahltest

für die Zahlen $z := 2$ bis $p - 1$ tue
    falls $z$ teilt $p$
        Teiler gefunden: $p$ ist keine Primzahl

falls keine Teiler gefunden
    $p$ ist Primzahl
sonst: $p$ ist keine Primzahl

Wie wird dieser Algorithmus in C implementiert?

*Euklidischer Algorithmus* berechne größten gemeinsamen Teiler zweier natürlicher Zahlen $p$ und $q$

1. Man dividiere $p$ ganzzahlig durch $q$. Dabei erhält man den Rest $r$, der zwischen 0 und $q - 1$ liegt.
2. Wenn $r = 0$ ist, dann ist $q$ der ggT. Wenn $r \neq 0$ ist, dann benenne das bisherige $q$ in $p$ um, das bisherige $r$ in $q$ und wiederhole ab Schritt 1.

| $p$ | $q$ | $r := p \bmod q$ |
|---|---|---|
| 216 | 378 | 216 |
| 378 | 216 | 162 |
| 216 | 162 | 54 |
| 162 | 54 | 0 |

$\Rightarrow \text{ggT}(216, 378) = 54$

Wie wird dieser Algorithmus in C implementiert?

## *Algorithmus*

- mechanisches Verfahren, das aus mehreren Schritten besteht
- Schritte werden sequentiell ausgeführt, bis das Ergebnis gefunden ist (es gibt auch parallele Algorithmen)
- einzelne Abschnitte des Verfahrens können mehrfach durchlaufen werden (Iteration, Schleife)

## *Entwurf von Algorithmen*

- finde eine Problemlösung
- formuliere sie in kleinen, elementaren Schritten

Was ist ein elementarer Schritt?

- Donald E. Knuth definiert eine Assembler-Sprache für einen fiktiven Computer MIX
- in der theoretischen Informatik arbeitet man mit Turing- und Register-Maschinen

$\rightarrow$ abhängig vom Kontext

Es gibt sehr alte, immer noch aktuelle Algorithmen:

- je zwei natürliche Zahlen haben einen ggT $\rightarrow$ Euklid
- Lösen linearer Gleichungssysteme $\rightarrow$ Gauß

$\rightarrow$ erst der Computer ermöglicht es uns, auch komplizierte Algorithmen mit tausenden von Schritten auszuführen

# Technische Informatik

Aufbau und Konstruktion von Computern.

- Rechnerarchitektur
- Rechnerhardware
- Mikroprozessortechnik
- Rechnernetze

# Praktische Informatik

Entwicklung und Erweiterung der Rechnereigenschaften.
Programmierung und Nutzung von Computern.

- Betriebssysteme
- Benutzerschnittstellen
- Informationssysteme (Datenbanken)
- Programmiersprachen und Übersetzer
- Softwaretechnologie

# Theoretische Informatik

Formale mathematische Grundlagen.

- Formale Sprachen
- Automatentheorie
- Berechenbarkeit
- Komplexitätstheorie
- Algorithmen & Datenstrukturen

# Angewandte Informatik

Lösen spezieller Probleme in Anwendungsbereichen mittels Computer. Der Rechner wird als Werkzeug eingesetzt.

- Computergrafik
- Digitale Signalverarbeitung (Bild-/Spracherkennung)
- Simulation und Modellierung
- Künstliche Intelligenz
- Textverarbeitung

# Anmerkungen

Praktische und Angewandte Informatik sind mitunter nur schwer abzugrenzen, weil in beiden die Programmierung im Mittelpunkt steht.

- *praktische Informatik:* die Eigenschaften des Rechners sollen erweitert werden, d.h. es soll uns eine bessere Maschine zur Verfügung stehen
- *angewandte Informatik:* spezielle Probleme in Anwendungsbereichen außerhalb der Informatik sollen gelöst werden
- heute wird dies durch eine andere Art der Einteilung ergänzt: *Wirtschafts-, Bio-, Geo-Informatik, ...*

# Anmerkungen

Informatik ist nicht gleichzusetzen mit Programmierung.

Man lernt Informatik nicht aus Büchern wie

- „Word 7.0 für Fortgeschrittene“ oder
- „Die besten Tipps zum Surfen im Internet“ oder
- „Programmieren in C++“.