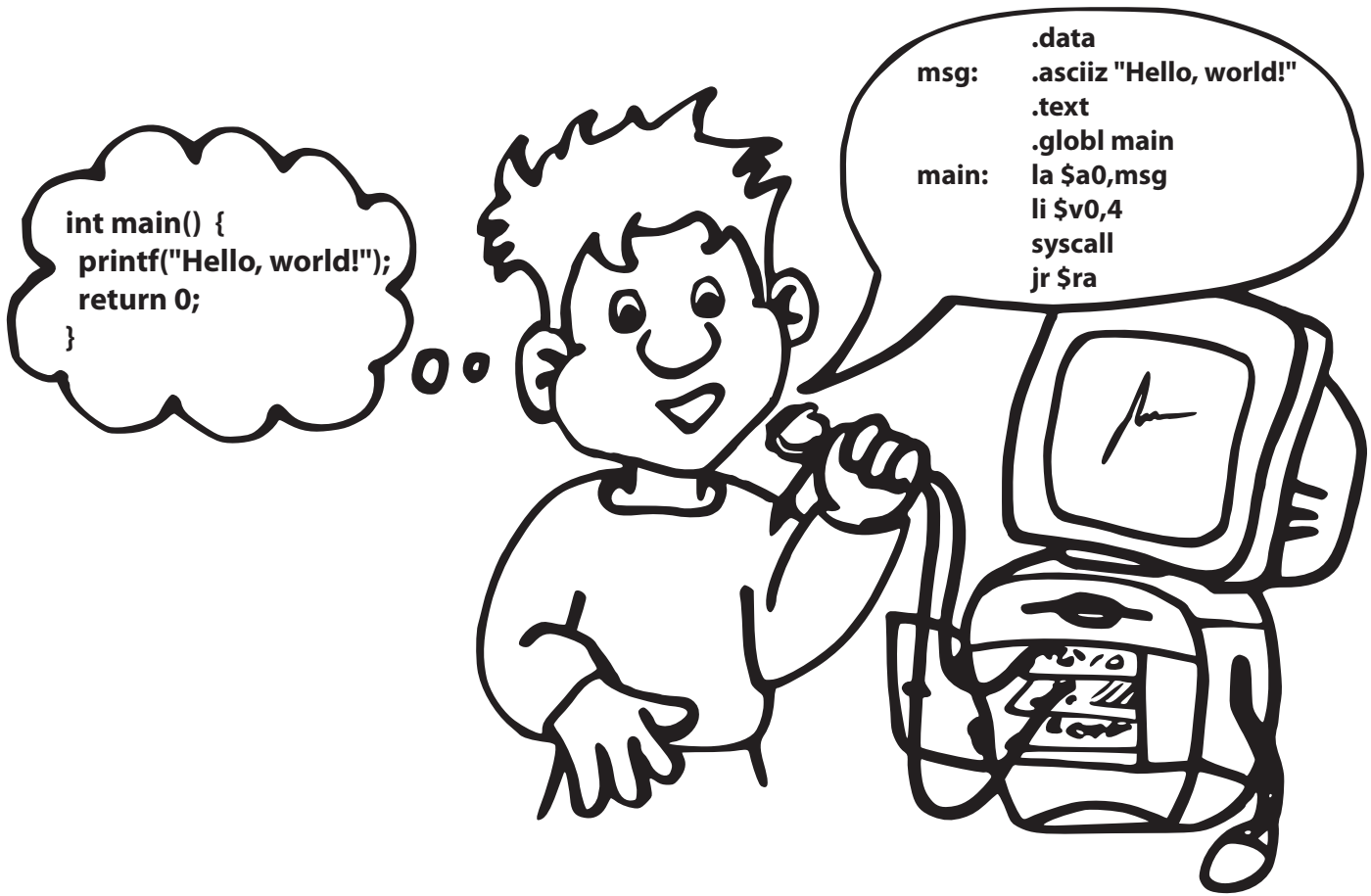
Grundlagen der Informatik III
Wintersemester 2010/2011 – 9. Vorlesung
B.Sc. Patrik Schmittat

TECHNISCHE UNIVERSITÄT DARMSTADT



Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

Literatur

TECHNISCHE UNIVERSITÄT DARMSTADT

[DBG08] Dausmann, Manfred, Ulrich Bröckl und Joachim Goll: C als erste Programmiersprache.
Teubner, 2008.

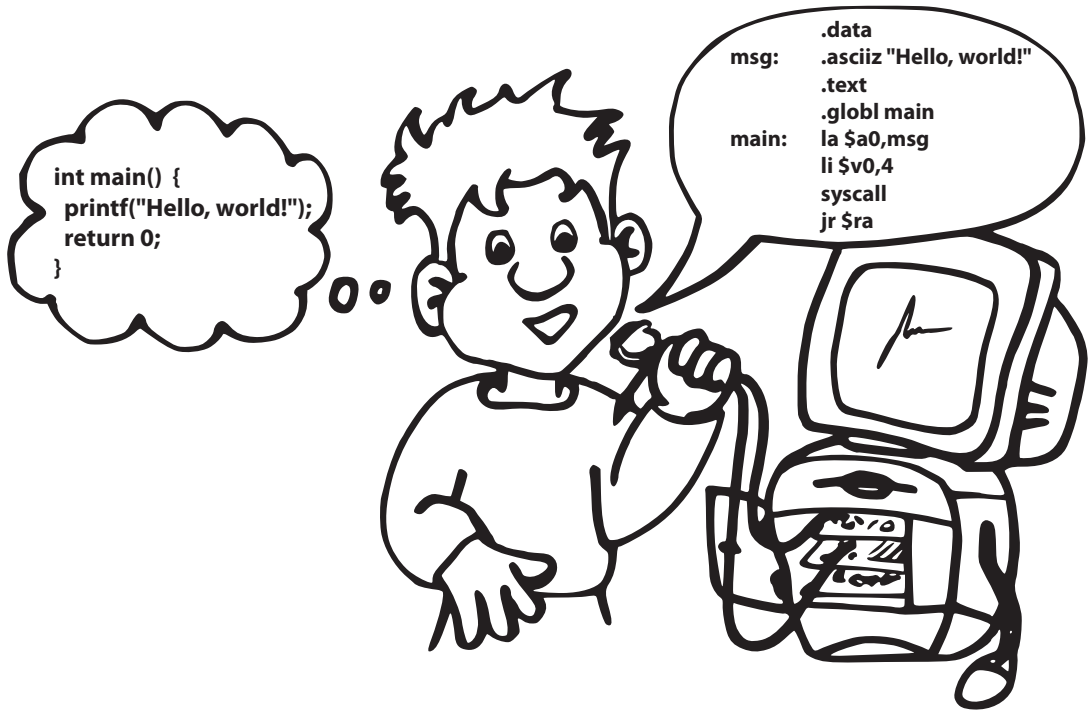
[Erl99] Erlenkötter, Helmut: C Programmieren von Anfang an.
Rowohlt Tb., 1999.

[Weba] The C++ Resources Network, `http://www.cplusplus.com/`.

[Webb] Your Resource for C and C++ Programming, `http://www.cprogramming.`

Einführung in C
Willkommen in C

TECHNISCHE UNIVERSITÄT DARMSTADT



- Von Assembler weg und hin zur Hochsprache
- Das ist das Ziel dieser Woche, oder anders: C in 2 Stunden ;)

Einführung in C
Zeitfaktor und Wartbarkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ihr habt gesehen, dass Assembler Kopfschmerzen verursachen kann
- Zeitfaktor ist wichtig und Assembler verschlingt viel davon
- 2 Minuten für den Code rechts aber 1,5 Stunden für die Assemblervariante
- Deswegen kommen wir nun zu den Hochsprachen

```
import java.util.Arrays;
public class BubbleSort {
  public static void main(String[] args) {
    int a[] = {16, 5, 18, 12, 11, 66, 62, 1};
    int n = 8;
    for (int i = n - 1; i > 0; i--) {
      for (int j = 0; j < i; j++) {
        if (a[j] > a[j+1]) {
          int temp = a[j];
          a[j] = a[j+1];
          a[j+1] = temp;
        }
      }
    }
    System.out.println(Arrays.toString(a));
  }
}
```

Einführung in C
Erstes C Programm

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#include <stdio.h>

main()
{
   printf("Hello World!\n");
}
```

- Simpelstes Programm zur Ausgabe von HelloWorld
- `#include` wird später behandelt
- `main()` Anfangspunkt für jedes Programm
- `printf(..)` Ausgabe eines Strings (später genauer)

Einführung in C
Struktur, Konventionen und Kommentare

TECHNISCHE UNIVERSITÄT DARMSTADT

- Whitespace[1] ist in C beliebig setzbar, aber trotzdem existieren Konventionen
- `{ .. }` werden in eine eigene Zeile gerückt
- Jede Deklaration / Definition ist von leeren Zeilen umgeben
- Jede Anweisung steht in einer eigenen Zeile
- Kommentare können mit `//` oder `/* .. */` eingefügt werden

[1] Darunter zählen Leerzeichen, Tabulatoren, Zeilenenden, etc.

Einführung in C
Kompilen und Linken

TECHNISCHE UNIVERSITÄT DARMSTADT

- Genau wie bei Assembler ist Kompilieren und Linken notwendig
- Für den Pool können folgende Befehle genutzt werden

```
gcc -o hello hello.c
./hello
```

- `-o hello` ist hier der Name der Binary
- Für den Anfang reicht ein normaler Texteditor aus
- Jedoch könnte `gdb` wichtig werden

Einführung in C
Typen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Genau wie in Assembler existieren Typen, hier nur explizit modelliert
- Hier eine kurze Auflistung der wichtigsten Typen

Typ	Größe	Bereich (signed, unsigned)
char	1 Byte	-128 - 127 0 - 255
short	2 Bytes	-32768 - 32767 0 - 65535
(long) int	4 Bytes	-2147483648 - 2147483647 0 - 4294967295
~~bool~~	~~1 Byte~~	~~true / false~~ Nur in C++
float	4 Bytes	$\pm$3.4e $\pm$38 (ca. 7 Nachkommastellen)
(long) double	8 Bytes	$\pm$1.7e $\pm$308 (ca. 15 Nachkommastellen)
wchar_t	2 / 4 Bytes	

Einführung in C
Deklaration, Variablen und Casten

TECHNISCHE UNIVERSITÄT DARMSTADT

- C ist CaseSensitive
- Bei den Namen CamelCase nutzen z.B. `ichBinEineVariable`

```
int a;
float b = 1.0f;
short c, d = 13, e;
unsigned int = 42;
char *str;
```

```
#include <math.h>

main()
{
  int p = 2, q = 4;
  float x1, x2, sqr;
  sqr = sqrt((p * p) / 4.0f - q);
  x1 = -p / 2.0f + sqr;
  x2 = -(float)p / 2 - sqr;
}
```

Einführung in C
Sichtbarkeit (Scope)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Variablen sind nur innerhalb ihrer { .. } sichtbar

```
main()
{
  int a, b;
  a = 0; // Ok
  {
    b = 0; // Ok
    int c = 0; // Ok
  }
  c = 0; // Fehler
  a = 0; // Ok
}

b = 0; // Fehler
```

Einführung in C
#include

TECHNISCHE UNIVERSITÄT DARMSTADT

- `#include` wird genutzt, um Dateien einzubinden
- Ist ein Präprozessor-Makro und es gibt auch andere (`#if #ifndef #endif`)
- In dem Sinne wird an dieser Stelle Copy / Paste durchgeführt
- Damit Header nicht zweimal eingebunden werden gibt es Include-Guards

```
#include <stdio.h> // Bindet stdio.h ein

#include <..> // Header auf Systemebene suchen
#include ".." // Header auf Projekt- dann Systemebene suchen

// Typischer Include-Guard der Header-Datei example.h
#ifndef EXAMPLE_H
#define EXAMPLE_H
.. hier der eigentliche Code ..
#endif // EXAMPLE_H
```

Einführung in C
Zeichenketten (Strings)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In dem Typ `char*` können Strings abgespeichert werden

```
#include <stdio.h>

main()
{
   char *str = "Hallo␣Welt!\n";
   printf(str);
}
```

- Dabei ist \n ein Steuerzeichen

\n	Zeilenumbruch (newline)	\b	Rückschritt (backspace)
\r	Wagenrücklauf (carriage return)	\a	Ton ausgeben
\t	Tabulator	\¨	Anführungszeichen (¨)
\v	Vertikaler Tabulator	\\	Schrägstrich (backslash) (\)

Einführung in C
Operatoren

TECHNISCHE UNIVERSITÄT DARMSTADT

+ − * /	Grundrechenarten
%	Modulo
++ −−	Preorder ++i == **return** i=i+1;
	Postorder i++ == **return** i; i=i+1;
& | ^ ~	Bitweise AND, OR, XOR, NOT
<< >>	Shift
== !=	(Un-)Gleich
> >= < <=	Größer / Kleiner
&& || !	Logisches AND, OR, NOT
&	Adresse von (monadisch)
*	Dereferenzierung (monadisch)
a ? b : c	Bedingter Ausdruck **if** (a) b; **else** c;

Einige der Operatoren sind auch mit = verwendbar z.B. +=

Einführung in C
Bindung

TECHNISCHE UNIVERSITÄT DARMSTADT

++ −−	Postorder
( )	Funktionsaufruf
[ ]	Arrayzugriff
++ −−	Preorder
!	Logisches NOT
~	Bitweise NOT
− +	unäres Minus / Plus
& *	Adresse / Dereferenzierung
(type)	Typumwandlung (cast)
−> .	Strukturzugriff
* /	Grundrechenarten
%	Modulo

+ −	Grundrechenarten
<< >>	Shift
> >= < <=	Größer / Kleiner
== !=	(Un-)Gleich
&	Bitweise AND
^	Bitweise XOR
|	Bitweise OR
&&	Logisches AND
||	Logisches OR
a ? b : c	Bedingter Ausdruck
=	(zgs) Zuweisung

Je höher ein Operator in der Liste ist, desto stärker bindet er
Mit Linien abgetrennte Operatoren besitzen die gleiche Bindung

Einführung in C
Kontrollstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

- `if else switch while do for` stehen auch in C zur Verfügung
- Sonderheiten:
  - Ausdruck im `if(..)` muss nicht bool sein (`exp != 0 === true`)
  - Falls `break;` im `switch` fehlt wird Code fortgeführt!
  - Deklaration der Schleifenvariable muss vor der `for` stehen

```
// String / Byte Copy
while(a[i++] = b[j++]);

switch(val)
{
  case 0: result = 0;
  case 1: result = 1;
} // result === 1
// Da kein break; vorhanden ist
```

```
int i; // Muss davor stehen!
for(i = 0; i < n; i++)
  // Befehle

int result = exp ? 1 : 0;
// ===
int result;
if(exp) result = 1;
else result = 0;
```

Einführung in C
Sprungbefehle break, continue

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schleifen können unterbrochen werden

```
while(true)
{
   if(exp)
      // Springt raus
      break;
}
```

- Während Schleifendurchlauf wieder an den Anfang springen

```
int i;
for(i = 0; i < n; i++)
{
   if(a[i] == 0)
      // Element ueberspringen
      continue;
   // Befehle
}
```

Einführung in C
Ein- und Ausgabe, Formatierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- `printf(..)` ist eine Methode um auf die Ausgabe zu schreiben
- Eingaben können mit `scanf` erledigt werden

```c
#include <stdio.h>

main()
{
  char str[80];
  int i;

  printf("Dein Name: ");
  scanf("%s", str);
  printf("Dein Alter: ");
  scanf("%d", &i);
  printf("%s %d\n",str,i);
}
```

```c
int printf(const char *format, ... );
```

%Frmt	Bedeutung	Beispiel
c	Buchstabe	'a'
d, i	Ganzzahl mit Vorzeichen	392
f	Gleitkommazahl	392.65
o	Oktahlzahl mit Vorzeichen	610
s	String	"sample"
u	Ganzzahl ohne Vorzeichen	7235
x, X	Hexzahl ohne Vorzeichen	7fa, 7FA
%	% ausgeben	

Einführung in C
Konvertierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Es stehen ein paar Funktionen zur Konvertierung zwischen `char*` und anderen Typen zur Verfügung
- Notwendig ist das Einbinden von `stdio.h`

```
int atoi(const char *str);
long atol(const char *str);
float atof(const char *str);
```

- Nicht im Standard aber vielleicht verfügbar
- Negative Zahlen gehen nur mit 10er-System

```
char *itoa(int value, char *str, int base);
```

- Standard-konform wäre die Nutzung von `sprintf`

```
sprintf(str, "%d", value) // 10er-System
sprintf(str, "%x", value) // 16er-System
sprintf(str, "%o", value) // 8er-System
```

Einführung in C
Mathematische Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Manche sind in `math.h` definiert

```
int abs(int val);
float fabs(float val);
double cos/sin/tan(double val);
double acos/asin/atan/atan2(double val);
double cosh/sinh/tanh(double val);
double exp/log/log10/sqrt(double val);
double pow(double base, double exp);
double ceil/floor(double val);
```

Einführung in C
String Operationen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Prinzipiell sind immer die Funktionen mit `n` sicherer
- Wichtig ist, dass `dest` den komplett erzeugten String halten können muss!

```
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t num);
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t num);
int strcmp(const char *str1, const char *str2);
int strncmp(const char *str1, const char *str2, size_t num);
char *strstr(char *str, const char *sub);
size_t strlen(const char *str);
```

Einführung in C
Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

- Arrays sind eigentlich wenig unterschiedlich zu Java
- Nur die dynamische Allokation ist anders
- Ein äquivalent zu Javas `.length` existiert nicht!

```
int i, n = 3;
int a[] = {4, 6, 1};
int b[5];
char century [100][365][24][60][60]; // >3GB Speicher ;)
for(i = 0; i < n; i++) a[i] = 10 - a[i];
func(a);
func(b); // Leider kein Kompilerfehler

void func(int vec[3]) { .. }
// Auch die gleiche Signatur
void func(int vec[]) { .. }
void func(int *vec) { .. }
```

Einführung in C
Structs, typedef

TECHNISCHE UNIVERSITÄT DARMSTADT

- Structs sind eine Art und Weise Daten zu koppeln
- Mit `typedef` können Typen umbenannt werden

```
struct typename { .. } objectname;

struct vector3_t
{
  int x, y, z;
}; // Es ginge auch hier a, b, c zu schreiben
vector3_t a, b, c;

a.x = 3; a.y = 2; c.z = 2;

typedef unsigned int myint;
myint i = 1;
```

Einführung in C
enum, #define, const

TECHNISCHE UNIVERSITÄT DARMSTADT

- `enum` dient dazu eine Auswahl von Werten zu erzeugen
- `const` deklariert sich niemals ändernde Variablen
- `#define` kann genutzt werden um Makros zu schreiben
- Jedoch sollte vorsichtig mit Makros umgegangen werden!

```
enum fruits_e
{
  APPLE,
  PEAR,
  BANANA
};

fruits_e myFruit = APPLE;
if(myFruit == PEAR) \\Befehle
```

```
// Im Scope immer 12
const monthsInYear = 12;
// Gleiche Bedeutung
#define MONTHSINYEAR 12

// Makros = boese!
#define SCARY ==0) printf
if(i SCARY ("Hilfe!");
```

Einführung in C
Funktionen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Deklarationen von Funktionen verhält sich prinzipiell nicht anders als sonst
- Wichtig ist aber, dass die Funktionen die man nutzen möchte über der aktuellen deklariert sein müssen!

```
// Deklarationen
void func(int i);
int func2(float a, double c);
// Definitionen
void func(int i)
{
  // Ginge ohne die Deklaration
  //   von func2 oben nicht!
  func2(2.0f, 1.0);
}
int func2(float a, double c) { .. }
void func3() { .. } // Deklaration und Definition zugleich
```

Einführung in C
Defaultwerte, Rekursion und Überladung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Defaultwerte für Parameter sind in C nicht verfügbar (nur in C++)
- Bei Rekursion kein anderes Verhalten
- Überladung funktioniert ebenso identisch

```
void func(int i /* = 0 funktioniert nicht */) { .. }

void rekfunc()
{
  // Ja die schoene 'Endlosrekursion'
  rekfunc();
}

// Ueberlaed die func(int) mit einem float-Parameter
void func(float f) { .. }
```

Zusammenfassung und Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erstes Programm in C
- Struktur, Konventionen und Kommentare
- Typen, Variablen, Cast, Scope
- include, Strings
- Operatoren, Bindung, Kontrollstrukturen
- Ein- / Ausgabe, Konvertierung
- Mathematische und String Funktionen
- Arrays, Structs, typedef, Enums, define, const
- Funktionen, Überladung, Rekursion

Nächste Vorlesung behandelt

- Zeiger, call-by-value / -reference
- malloc, free, calloc, realloc
- Lokale, globale, statische Variablen
- Datei Ein- / Ausgabe
- Argumente, gdb, Dokumentation