

Tutorium 1: „Die Welt da draußen“ Input, Output, Dateien

Funktionsreferenzen

- <http://www.cplusplus.com/reference/cstdio>
- man stdio
- man printf
- man scanf
- man fopen
- man ey was halt sonst noch so vorkommt

Gliederung

- Ausgabe mit printf
- Eingabe mit scanf
- Dateiein- und Ausgabe

Ausgabe mit printf

- Bereits aus Hello World Programm bekannt
printf(„Hello World\n“);
- Prototyp: *int printf (const char * format, ...);*
 - Parameter:
 - Formatstring *format*
 - beliebig viele weitere Parameter jeden Basistyps
 - Rückgabe:
 - Success: Anzahl geschriebener Zeichen (≥ 0)
 - Error: Negativer Wert

Ausgabe mit printf

- Formatstring: Häh?
 - kann normalen Text wie „Hello World“ enthalten
 - kann in diesem Text Formatzeichen enthalten
 - jedes Formatzeichen entspricht einem nach dem Formatstring übergebenem Parameter, der an der Stelle des Formatzeichens im Text ausgegeben werden soll

- Beispiele:

Formatzeichen	Dargestellter Datentyp
%d	Integer
%f	Float (Fließkommazahl)
%s	String
%x	Hexadezimale Ausgabe

Ausgabe mit printf

Beispiel printfDataTypes.c

Ausgabe mit printf

- Das geht doch schöner!
- Vollständige Syntax eines Formatzeichens:
%[flags][width][.precision][length]specifier
- *flags*: Verschiedene Flags (Überraschung)
- *width*: Minimale Breite des Ausgabefeldes
- *precision*: Genauigkeit mit der der Wert ausgegeben werden soll (Achtung: unterschiedliche Bedeutung für versch. Typen)
- *length*: Angabe über Datentypgröße

Ausgabe mit printf

Beispiel printfFormatting.c

Eingabe mit scanf

- printf, nur umgedreht - oO?
- Prototyp: *int scanf (const char * format, ...);*
 - Parameter:
 - Formatstring *format*: Enthält jetzt Formatzeichen zur Information darüber was für Werte eingelesen werden sollen (Buchstaben genau wie printf, aber andere Syntax)
 - beliebig viele weitere Variablen**adressen**
 - Rückgabe:
 - Success: Anzahl der gelesenen Werte (≥ 0)
 - Error: EOF (vordefinierte Konstante für End Of File)

Eingabe mit scanf

- Syntax des Formatstrings:

%[][width][modifiers]type*

- *: Gelesener Wert wird nicht gespeichert
- *width*: Maximale Zeichenanzahl die für den Wert eingelesen werden darf - wichtig bei Strings
- *modifiers*: Gibt Datentypgröße an
- *type*: Die Datentypen bekannt aus printf, also z.B. d für Integer, f für Float etc.

Eingabe mit scanf

Beispiel scanfDataTypes.c

Eingabe mit scanf

- wenn normaler Text im Formatstring enthalten ist, wird dieser als Vergleich mit der Eingabe benutzt
 - weicht die Eingabe ab, wird abgebrochen
 - ist sie korrekt werden alle Parameter eingelesen

Eingabe mit scanf

Beispiel `scanfNonFormats.c`

Dateien

- Auch in C gilt: Dateien kann man
 - Öffnen mit `fopen(...)`
 - Schliessen mit `fclose(...)`
 - Wahnsinn!

Dateien

- Prototyp `fopen`:

```
FILE * fopen ( const char * filename,  
const char * mode );
```

- Parameter:

- Dateiname *filename* als String
- Dateimodus *mode* als String

- Rückgabe:

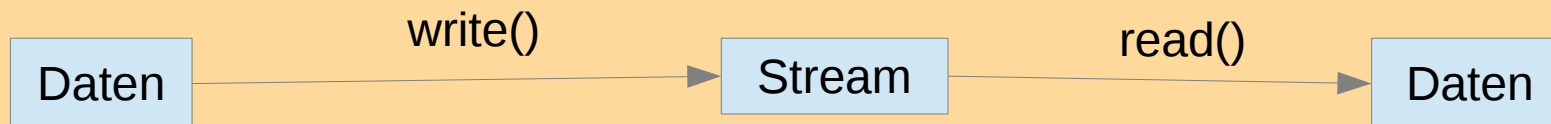
- Success: **Stream** (Dateideskriptor), kann für weitere Operationen benutzt werden
- Error: NULL (vordefinierte Konstante)

Dateien

- Mögliche Dateimodi:
 - „r“: Datei zum Lesen öffnen
 - „w“: Datei zum (Über-)Schreiben öffnen
 - existiert sie noch nicht, wird sie angelegt
 - existiert sie, wird bisheriger Inhalt überschrieben
 - „a“: Datei zum Anhängen öffnen
 - existiert sie noch nicht, wird sie angelegt
 - existiert sie, werden geschriebene Daten am Ende angehängt

Dateien

- Moment. Was ist ein Stream?
 - abstraktes Objekt zur Kommunikation
 - (wichtigsten) mögliche Operationen darauf:
 - read
 - write
 - Daten werden in geordneter Reihenfolge gelesen und geschrieben (exakt: FIFO)



Dateien

- Vorteile von Streams:
 - bereits vom Betriebssystem implementiert
 - Standardstreams (Linux):
 - stdout: Standard Output (dorthin gibt printf() Daten aus)
 - stdin: Standard Input (von dort liest scanf() Daten)
 - stderr: Standard Error (Fehlerausgabe)
 - einfache Handhabung
 - Sehr einfach umleitbar (zb. Piping in der Konsole)
 - Auch Dateien können als Streams behandelt werden

Dateien

Beispiel `pipingOutput.c`
(Piping von `pipingOutput` nach `scanfNonFormats`)

Dateien

- Prototyp fclose:

```
int fclose ( FILE * stream );
```

- Parameter
 - Stream *stream*, der vorher mit `fopen()` geöffnet wurde
- Rückgabe
 - Success: 0
 - Error: EOF

Dateien

Beispiel filesOpenClose.c

Dateien

- Jetzt der Clou (Trommelwirbel): Für Dateiein- und Ausgabe benutzen wir einfach `fprintf()` und `fscanf()` wie `printf()` und `scanf()`, nur dass wir nicht `stdin` oder `stdout` benutzen, sondern den Stream den wir durch `fopen()` erhalten haben
- *`int fprintf (FILE * stream, const char * format, ...);`*
 - Zusätzlicher *stream* Parameter
- *`int fscanf (FILE * stream, const char * format, ...);`*

Dateien

Beispiel filesInput.c

Dateien

Beispiel filesOutput.c

Dateien

- Weitere nützliche Funktion: feof()

*int feof (FILE * stream);*

- Parameter:
 - Bekannter Stream *stream* von fopen()
- Rückgabe
 - End Of File reached: non-Zero
 - End Of File not reached: zero

Dateien

Beispiel filesInputEverything.c

Fragen / Anmerkungen / Kritik / Applaus

Zugabe: Konsolenparameter

- Beliebt sind auch Parameter, die an das Programm auf der Konsole übergeben werden
./programm parameter1 parameter2
- Benutzen kann man diese als Parameter der main-Funktion

Zugabe: Konsolenparameter

- Syntax der main-Funktion mit Parametern:

```
int main(int argc, char *argv[ ]);
```

- argc enthält die Anzahl der Parameter die im String-Array argv gespeichert sind
- argv[0] enthält den Programmnamen
- argv[1...argc-1] enthalten weitere Parameter

Zugabe: Konsolenparameter

Beispiel consoleParameters.c

Nu is aber Schluss.