Programmieren in C

SS 2019

Vorlesung 11, Dienstag 16. Juli 2019

(Optionen)

Prof. Dr. Peter Thiemann
Lehrstuhl für Programmiersprachen
Institut für Informatik
Universität Freiburg

Teile der Folien enthalten Material von Prof. Dr. Bast

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Erfahrungen mit dem Ü10 — Life & JSON
- Inhalt
  - Globale Variablen — Deklaration mit "extern"
  - Kommandozeilenoptionen — getopt_long

Projekt: vier gewinnt

Erfahrungen mit dem Ü10 1/3

UNI FREIBURG

- Zusammenfassung / Auszüge
  - Vorlage lief nicht durch checkstyle…
    - Abhilfe: [whitespace/parens]-Fehler in cpplint ausschalten (siehe Forum letzten Dienstag)
    - Konnte leider erst Freitag reparieren
  - "Mich hat etwas gestört, dass die Codevorlage aus der Vorlesung voller checkstyle-Fehler war."
  - "Was allerdings sehr nervig war, war die Tatsache das die Dateien von der Vorlesung für Json mit mehr als 100 Checkstyle Fehler versehen waren."

Erfahrungen mit dem Ü10 2/3

UNI FREIBURG

- Einzelkommentare
  - "sehr gute Vorlesung, interessantes Thema."
  - "Super Aufgabe!!" und "Schöne Aufgabe, gut zu bearbeiten." (jeweils zum Game of Life)
  - "War gut, netteres Blatt.. Saß trotzdem wie immer länger dran, aber ich persönlich mochte das Knobeln ja bei jedem Blatt (Außer Keilschrift!!;))"
  - "Die Vorlesung fand ich interessant."
  - "Zum ersten Mal hatte ich Spaß an den Aufgaben"

Erfahrungen mit dem Ü10 3/3

UNI FREIBURG

- Zeitbedarf (Stand 8:30)
  - 2 * 2h
    1 * 4h
    1 * 5h
    5 * 7h
    6 * 8h
    2 * 9h
    1 * 12h
    1 * 14h
    2 * 15h
  - N=21, Durchschnitt 8.1h, Median 8h

Globale Variablen 1/4

UNI FREIBURG

- Was + warum
  - Variablen, die außerhalb einer Funktion definiert sind, nennt man **globale Variablen**

```
int x;

void someFunction() {
  // x can be used here.
  ...
}
```

  - Globale Variablen können überall im Code benutzt werden, auch in anderen Dateien, aber es sollte nach Möglichkeit vermieden werden
  - Ausnahme: mit static vereinbarte Variable sind nur in der gleichen Datei sichtbar

Globale Variablen 2/4

UNI FREIBURG

- Deklaration von globalen Variablen: Wie Funktionen
  - Jede globale Variable muss vor der Benutzung deklariert werden

    Üblicherweise in einer .h Datei, die dann inkludiert wird, wenn die Variable benötigt wird
  - Jede globale Variable muss in genau einer Datei implementiert sein

    Die dazugehörige .o Datei oder Bibliothek muss beim Linken dabei sein

Globale Variablen 3/4

UNI FREIBURG

- Deklaration von globalen Variablen
  - Mit dem Schlüsselwort extern

```
extern int x;
extern int y;

int main(int argc, char** argv) { ... }
```

  - Die Variablen x und y müssen in genau einer anderen Datei definiert werden:

```
int x;
int y;
```

  - Wenn eine globale Variable mit extern deklariert wurde und dann beim Linken nicht gefunden wird, gibt es einen Fehler "undefined reference to ..."

Globale Variablen 4/4

UNI FREIBURG

- Verwendung von globalen Variablen
  - Möglichst nicht verwenden!
  - Manche Bibliotheken verwenden sie als Rückgabewerte und/oder zum Speichern von Zwischenzuständen
    - Beispiel: getopt()
  - Viele Funktionen aus der Standardbibliothek liefern Fehlercodes in der globalen Variable errno
    - (Die Funktion perror() druckt dazu die Standardnachricht auf stderr aus.)
    - Header #include <errno.h> enthält Makros der Fehlercode und Fehlernamen

Parsen von Optionen 1/6

UNI FREIBURG

- Beispiel mit "langen" Optionennamen
  - Typischer Aufruf von der Kommandozeile

    ./InputOutputMain --head=3 --numbers example.csv
  - Zur Erinnerung: Argumente der main Funktion

    int main(int argc, char** argv);
  - Die Werte von argv sehen dann so aus:

    argv[0] : "./InputOutputMain"
    argv[1] : "--head=3"
    argv[2]: "--numbers"
    argv[3] : "example.csv"

Parsen von Optionen 2/6

UNI FREIBURG

- Beispiel mit "kurzen" Optionennamen
  - Äquivalenter Aufruf zu dem von der Folie vorher:

    ./InputOutputMain -h 3 -n example.csv
  - Argumente der main Funktion

    int main(int argc, char** argv);
  - Die Werte von argv sehen jetzt so aus:

    argv[0] : "./InputOutputMain"
    argv[1] : "-h"
    argv[2] : "3"
    argv[3]: "-n"
    argv[4] : "example.csv"

Parsen von Optionen 3/6

- Verarbeitung mit getopt, Teil 1/3
  - Im Programm muss ein Feld von Strukturen definiert werden, in dem die Optionen definiert werden (langer Name, Status des Arg., NULL, kurzer Name)

```
#include <getopt.h>

struct option options[] = {
   { "head", required_argument, NULL, 'h' },
   { "numbers", no_argument, NULL, 'n' },

   { NULL, 0, NULL, 0 }              // End of array.
};
```

  - An der dritten Position kann statt NULL ein Zeiger auf eine Variable (Typ int*) stehen

    Siehe "man 3 getopt" für die Semantik davon

Parsen von Optionen 4/6

UNI FREIBURG

- Verarbeitung mit getopt, Teil 2/3
  - Verarbeiten der Optionen:

```
optind = 1;                          // Start with argv[1].
while (true) {
   // s and n = short names, the : means with argument.
   char c = getopt_long(argc, argv, ”h:n", options, NULL);
   if (c == -1) break;               // No more options.
   switch (c) {                      // c is the short name.
   case 'h’: head = atoi(optarg);    // Argument in optarg.
             break;
   case 'n': numbers = true;         // Option without arg.
  }
}
```

Parsen von Optionen 5/6

UNI FREIBURG

- Verarbeitung mit getopt, Teil 3/3
  - Jetzt noch die Nicht-Options Argumente

    ```
    if (optind + 1 != argc) { printUsageAndExit(); }
    fileName = argv[optind];
    ```

  - Achtung: optind zeigt nach der vorherigen Schleife auf das nächste Argument, das keine Option mehr ist
  - Das funktioniert, weil die Schleife auf der Folie vorher die Optionen und Ihre Argumente "nach vorne tauscht"

    Vorher: ./InputOutputMain –h 3 example.csv –n
    und optind == 1, so dass argv[optind] == "-h" ist

    Nachher: ./InputOutputMain –h 3 –n example.csv
    und optind == 4, so dass argv[optind] == "example.csv"

Parsen von Optionen 6/6

UNI FREIBURG

- Zwei wichtige globale Variablen aus getopt.h
  - **optind** ist der Index von dem Argument, das getopt_long als nächstes bearbeitet

    optind ist zwar mit 1 initialisiert, aber Achtung:

    beim Testen wird die getopt_long Schleife evtl. mehrmals hintereinander ausgeführt, deswegen vorher immer optind = 1 setzen
  - **optarg** ist das Argument der zuletzt verarbeiteten Option, sofern sie ein Argument hatte (sonst NULL)

    optarg ist immer vom Typ char*; bei Bedarf selbst

    konvertieren

Hinweise zum Testen 1/1

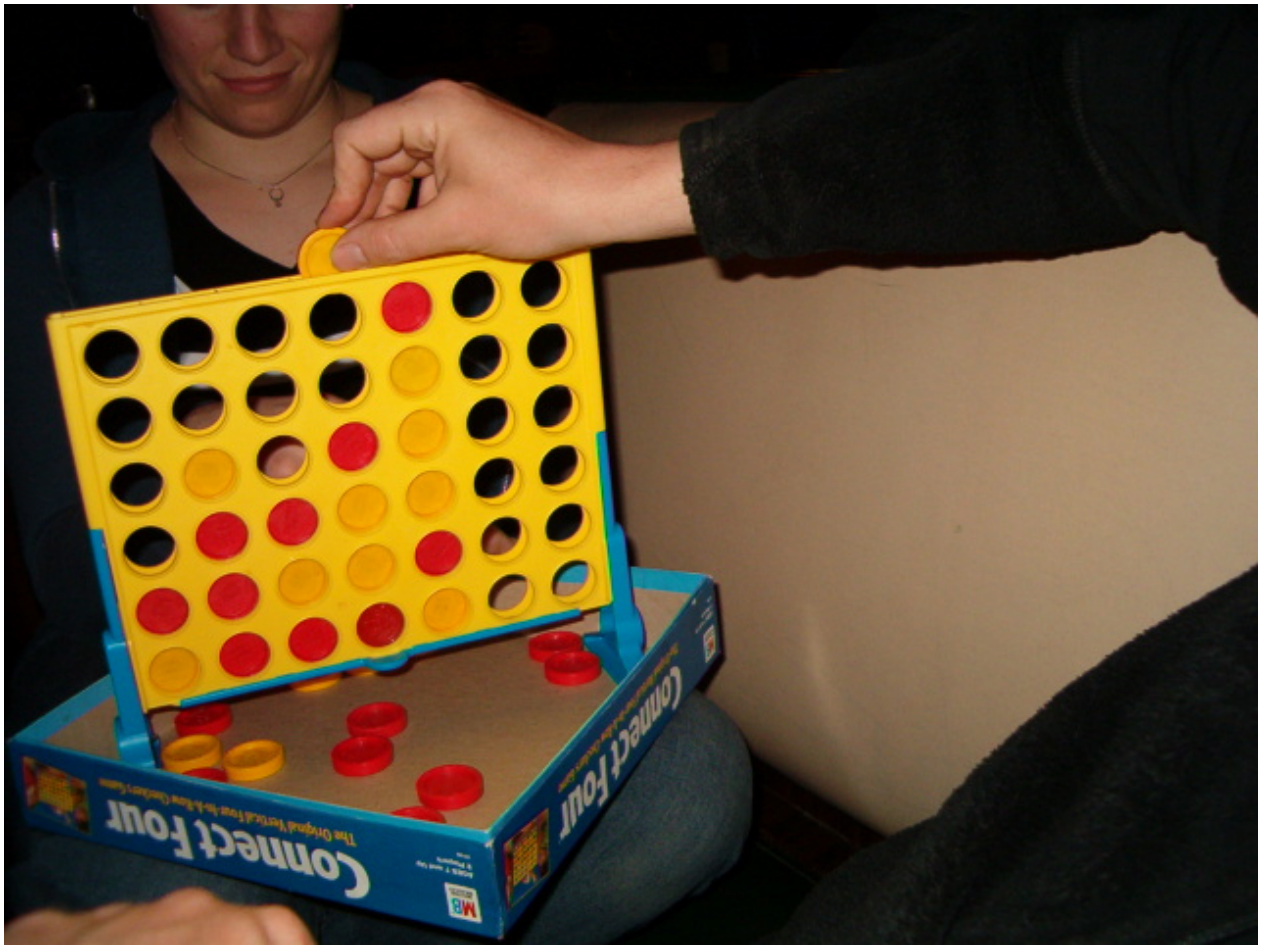
UNI FREIBURG

- Kommandozeilenparser
  - Das Parsen der Kommandozeilenparameter grundsätzlich auch testen
  - Für den Test Case muss **argv** explizit gesetzt werden:

```
char* argv[2] = { "arg1", "arg2" };
```

Hinweise zum Projekt 1/5

- Vier gewinnt



Hinweise zum Projekt 2/5

UNI FREIBURG

- Struktur
  - Basis Aufgabe 0 (30): verpflichtend
  - Alternative Zusätze
    - Alternative 1 (50): Graphische Bedienung
    - Alternative 2 (50): einfacher Computerspieler
  - Erweiterung 3 (20): Laden/Abspeichern von Spielständen
- Volle Punktzahl: 80
- Abgabe 10. August

Hinweise zum Projekt 3/5

UNI FREIBURG

- Korrekturschlüssel wie bekannt
  - 50% korrekte Funktion
  - 25% Testen (außer bei Graphik)
  - 25% Coding style
    - Kommentieren, Strukturierung, Beachtung der Warnungen bzgl Speicherallokation, Schleifen, etc
- Wichtig
  - Alle Funktionen kommentieren
  - Insbesondere auch die Tests kommentieren
    - Beispiel: bei einem Test des Computerspielers muß klar erklärt werden, welche Eigenschaft getestet wird

Hinweise zum Projekt 4/5

UNI FREIBURG

- Zeitfallen vermeiden
  - A1/A2 je ca 1h anschauen, dann für eine entscheiden
  - Gold-Plating vermeiden:
    - Ausgefeiltes UI, Fancy Graphik
    - Textausgabe in der Graphik
    - Perfekte AI
  - Stattdessen
    - Funktion richtig hinbekommen
    - Innere Werte
    - So strukturieren, dass sinnvoll getestet werden kann

Hinweise zum Projekt 5/5

UNI FREIBURG

- Zum Computerspieler (vgl wikipedia)
  - Ein Spiel mit perfekter Information
  - Vollständig gelöst in 1988/1990
  - Ergebnis
    - Der erste Spieler kann das Spiel gegen beste Verteidigung gewinnen, wenn er in der mittleren Spalte beginnt.
    - Beginnt er in der Spalte links oder rechts daneben, endet das Spiel bei beiderseits perfektem Spiel remis;
    - wirft er seinen ersten Stein in eine der vier restlichen Spalten, verliert er gegen einen perfekten Gegner sogar.

Literatur / Links

UNI FREIBURG

- Parsen von Optionen
  - man 3 getopt