Programmieren in C

SS 2019

Vorlesung 10, Dienstag 2. Juli 2019

(Parsing, Enums, Unions)

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

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Erfahrungen mit dem Ü8 — intplane + Dateien
  - Punkte aus der Evaluation — Erinnerung + Klarstellung
- Inhalt
  - Recursive Descent Parsing — Rezept + Beispiel JSON0
  - Repräsentation von Bäumen — enum, union
  - **Ü9: weiterer Aufbau für Game of Life, Ein-/Ausgabe mit Dateien**

Erfahrungen mit dem Ü8 1/3

UNI FREIBURG

- Einzelkommentare
  - "Die Aufgaben waren dieses Mal interessant und machbar."
  - "Die Vorlesung war leider wieder ziemlich unverständlich, vor allem der zweite Teil über GLES“

  Heute nochmal Gelegenheit nachzufragen. Nächste Woche kommt eine Aufgabe dazu.

  GLES: Wo beginnt der Aussetzer?

  - "Der Schwierigkeitsgrad der Aufgaben ist angemessen."
  - "Würde mir wünschen wenn man schon in der 1. Übung eine VM bekommt, mit der die folgenden Übungen auszuführen sind.“

  Nächstes Mal…

Erfahrungen mit dem Ü8 2/3

UNI FREIBURG

- Einzelkommentare
  - "Die Aufgaben waren teilweise mal wieder nicht gut beschrieben.“
  - "Das Übungsblatt hat die Vorlesung gut ergänzt."
  - "Aufgaben sehr ungenau gestellt."
  - "I am excited for the final project and applying openGL there."

Erfahrungen mit dem Ü8 3/3

UNI FREIBURG

- Bearbeitungszeiten / Statistik
  - Ü8, 22 Erfahrungen
  - Durchschnitt 9.1h, Median 9h
  - Zwischen 3h und 17h, 3 Personen >= 15h

  - Letztes Mal (Ü7): 5 Personen >= 15h

    Angebot steht immer noch:

    uns mal kontaktieren, wo das Problem liegt

Prüfungstechnisches 1/1

UNI FREIBURG

- Wiederholung aus Vorlesung #1
  - Sie bekommen wunderschöne Punkte, maximal 16 pro Übungsblatt, das sind maximal 176 Punkte für Ü0 – Ü10
  - Für das Projekt gibt es maximal 80 Punkte
  - Macht insgesamt 256 Punkte
  - **Außerdem:** Zum Bestehen müssen mindestens 88 Punkte in den Übungen (Ü0 – Ü10) und mindestens 40 Punkte im Projekt erreicht werden
  - Für das Ausfüllen des Evaluationsbogens am Ende gibt es +16 Punkte. Davon können **jeweils 8** auf ein Defizit in Übung oder Projekt angerechnet werden.
  - **Außerdem 2:** Sie müssen sich mindestens einmal mit Ihrem Tutor / Ihrer Tutorin treffen (scheint zu laufen)

Parsing 1/8

UNI FREIBURG

- Erinnerung: JSON
  - Mensch- und Maschinenlesbares Datenaustauschformat
  - Einfaches Beispiel

```
{ "fruit": "Apple", "size": "Large", "color": "Red" }
```

  - Geschachteltes Beispiel

```
{ "host": "localhost",
  "port": 3030,
  "public": "../public/",
  "paginate": { "default": 10, "max": 50 },
  "mongodb": "mongodb://localhost:27017/api"
}
```

Parsing 2/8

- Wiederholung: Definition JSON0 (BNF Format)

```
Object ::=
    '{' '}'
    '{' Members '}'
Members ::=
    Member
    Member ',' Members
Member ::=
    String ':' Value
Value ::=
    Number
    String
    Object
```

Object, Members, Member, String, Value, Number sind Variable

'{' '}' ',' ':' sind Terminale: Zeichen, die so in der Eingabe vorkommen müssen

Übereinanderstehende Zeilen sind Alternativen

Parsing 3/8

UNI FREIBURG

- BNF (Backus-Naur Form; Backus-Normal Form)
  - Benannt nach John Backus (Turing Award 1977) und Peter Naur (Turing Award 2005) zur Beschreibung der Syntax von ALGOL60, einem Vorgänger von C
  - Fast jede Sprache hat heutzutage eine Beschreibung in BNF
  - Variable werden definiert durch Regeln der Form

```
Object ::=
    '{' '}'
    '{' Members '}'
```

  - Ein Vorkommen einer Variable wird durch eine der rechten Regelseiten ersetzt.
  - Diese Ersetzung geht solange weiter bis nur noch Symbole ohne Regeln vorhanden sind.

Parsing 4/8

UNI FREIBURG

- Beispiel

Object

-> '{' Members '}'

-> '{' Member '}'

-> '{' String ':' Value '}'

-> '{' String ':' Number '}'

  - Die jeweils ersetzte Variable ist in rot angezeigt
  - Zum Schluss nur noch Terminalsymbole und primitive Variablen ohne Regeln
  - Die muss ein Parser erkennen und einlesen

Parsing 5/8

UNI FREIBURG

- Lesen der primitiven Variablen -- String

  bool parse_string(FILE * f)

  - Vorbedingung: Datei ist bereit zum Lesen
  - Überlese Leerzeichen
  - Prüfe das nächste Zeichen auf “
  - Misserfolg und Rückstellung des Zeichens, falls es ein anderes ist
  - Sonst lies Zeichen bis “ oder Dateiende

Parsing 6/8

UNI FREIBURG

- Lesen der primitiven Variablen -- Number

  bool parse_number(FILE * f)

  - Vorbedingung: Datei ist bereit zum Lesen
  - Überlese Leerzeichen
  - Prüfe das nächste Zeichen auf Ziffer
  - Misserfolg und Rückstellung, falls ein anderes Zeichen gefunden
  - Sonst lies Ziffern bis keine weitere Ziffer folgt

Parsing 7/8

UNI FREIBURG

- Funktionsweise der Primitiven
  - Alle nach dem gleichen Muster
  - Betrachte das nächste Nicht-Leerzeichen
  - Melde Misserfolg und stelle das Zeichen zurück, falls es nicht passend ist
  - Dieses Muster hilft bei der Implementierung von Variablen mit mehreren Alternativen, wie Value

  bool parse_value(FILE * f)

  - Prüft die Alternativen der Reihe nach
  - Liefert eindeutiges Ergebnis, weil jede Alternative mit einer anderen Art Zeichen beginnt

Parsing 8/8

UNI FREIBURG

- Generelle Vorgehensweise
  - Jede Variable entspricht einer (rekursiven) Funktion
  - Alle Funktionen müssen vorab deklariert werden
  - Auf der rechten Regelseite
    - Jedes Vorkommen einer Variable = Aufruf der Funktion
    - Jedes Vorkommen eines Terminalsymbols = Aufruf von expect_char() mit dem entsprechenden Symbol
  - Rückgabewerte müssen auf Misserfolg getestet werden
  - Bei Alternativen beginne mit der ersten und springe bei Misserfolg am Anfang jeweils zum nächsten

Bäume 1/6

UNI FREIBURG

- Am Beispiel von JSON0
  - Ein JSON0 Objekt enthält Members, nämlich eine Liste von Member
  - Ein JSON0 Member besteht aus
    - Einem Attributnamen (String) und
    - dem zugehörigen Wert (Value)
  - Ein JSON0 Wert ist entweder
    - Ein String
    - Eine Zahl (Number) oder
    - ein Objekt (Object)

Bäume 2/6

UNI FREIBURG

- Member wird durch ein **struct** dargestellt
  - Ein JSON0 Member besteht aus
    - Einem Attributnamen (String) und
    - dem zugehörigen Wert (Value)

```
struct member_ {
  char const * member_name;
  value_t * member_value;
};
```

  - Ein „Name-Wert Paar" (name value pair)

Member_name	Member_value

Bäume 3/6

UNI FREIBURG

- Ein Objekt wird durch ein **struct** dargestellt
  - Ein JSON0 Objekt enthält Members, nämlich eine Liste von Member

```
struct object_ {
  object_t * members_next;
  member_t * members_member;
};
```

  - Next-Zeiger implementiert eine verkettete Liste
  - Member zeigt auf das aktuelle Name/Wert Paar

Members_next	Members_member

Bäume 4/6

UNI FREIBURG

- Für einen Wert gibt es mehrere Alternativen
  - Ein JSON0 Wert ist entweder
    - Ein String
    - Eine Zahl (Number) oder
    - ein Objekt (Object)
  - Wir müssen uns also merken, welche Art von Wert vorliegt. Das geht in C mit einem **enum** (Enumeration, Aufzählung) Typ.

```
enum type_ { Tnumber, Tstring, Tobject };
```

  - Definiert einen Typ enum type_ mit genau drei Werten: Tnumber, Tstring, Tobject
  - Beispiel:

```
enum type_ value_kind = Tnumber;
```

Bäume 5/6

UNI FREIBURG

- Abspeichern der Alternativen
  - Ein JSON0 Wert ist entweder
    - Ein String
    - Eine Zahl (Number) oder
    - ein Objekt (Object)
  - Wir brauchen einen Speicherbereich, der groß genug ist, dass ein String oder eine Zahl oder ein Objekt(zeiger) abgelegt werden kann.
  - Das geht mit einem **union** Typ.

Value_number
Value_string
Value_object

Bäume 6/6

UNI FREIBURG

- Abspeichern der Alternativen
  - Ein JSON0 Wert ist ein String, eine Zahl oder ein Objekt

```
struct value_ {
  enum type_ value_kind;
  union {
    int value_number;
    char const * value_string;
    object_t * value_object;
  } value_u;
} my_value;
```

  - Zugriff auf die Alternativen der union (hier die Zahl) erfolgt wie bei struct mit my_value.value_u.value_number
  - Aber nur nach test my_value.value_kind == Tnumber!

Literatur / Links

- Enums

https://www.geeksforgeeks.org/enumeration-enum-c/

- Unions

https://www.geeksforgeeks.org/union-c/

- JSON

http://json.org/