Programmiertechnik II

Sommersemester 2022 – Aufgabenblatt 2

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 2

Der zu verwendende Programmrahmen steht als Zip-Archiv `pt2_ex2_framework.zip` im Moodle unter der Adresse `https://moodle.hpi.de/course/view.php?id=297` zum Download zur Verfügung.

Aufgabe 2.1: WHILE-Programme (9 Punkte 2+1+2+1+3)

Algorithmen können in Form sogenannter LOOP- oder WHILE-Programme angegeben werden. Bearbeiten Sie die folgenden Aufgaben **schriftlich** unter Nutzung der in Abbildung 1 dargestellten Syntax. x_i und x_j bezeichnen Variablen, die an dieser Stelle eingesetzt werden, c einen konstanten Wert. „$P; P$“ ist die Ausführung zweier Teilprogramme hintereinander, „LOOP x_i DO P END“ steht für die x_i-fache Ausführung des Teilprogramms P.

$$
\begin{array}{rcl}
P & ::= & x_i := x_j + c \\
& | & x_i := x_j - c \\
& | & P; P \\
& | & \text{LOOP } x_i \text{ DO } P \text{ END}
\end{array}
$$

(a) LOOP-Programm

$$
\begin{array}{rcl}
P & ::= & x_i := x_j + c \\
& | & x_i := x_j - c \\
& | & P; P \\
& | & \text{LOOP } x_i \text{ DO } P \text{ END} \\
& | & \text{WHILE } x_i \neq 0 \text{ DO } P \text{ END}
\end{array}
$$

(b) WHILE-Programm

Abbildung 1: Syntax der LOOP- und WHILE-Programme

a) Zeigen oder widerlegen Sie: Ein LOOP-Programm terminiert immer nach endlicher Zeit.

b) Zeigen oder widerlegen Sie: Ein WHILE-Programm terminiert immer nach endlicher Zeit.

c) Zeigen oder widerlegen Sie: Alle LOOP-Programme können durch ein äquivalentes WHILE-Programm (ohne Verwendung von LOOP) simuliert werden.

d) Zeigen oder widerlegen Sie: Alle WHILE-Programme können durch ein äquivalentes LOOP-Programm simuliert werden.

e) Geben Sie drei WHILE-Programme ohne Verwendung von LOOP an, die die folgenden Funktionen berechnen. Dafür darf davon ausgegangen werden, dass gilt $x, y \in \mathbb{N}, x \geq 0, y > 0$.

   i) $x + y$

   ii) $x \cdot y$ (*Hinweis*: Es kann das Programm aus i) in der Form $P_+(x, y)$ benutzt werden.)

   iii) x^y (*Hinweis*: Es kann das Programm aus ii) in der Form $P_\cdot(x, y)$ benutzt werden.)

Hinweis: Zuweisungen von Konstanten dürfen als $x_i := c$ notiert werden. Geben Sie jeweils an, welche Variable das Ergebnis der Berechnung enthält, nachdem das Programm terminiert. Achten Sie gegebenenfalls auf Seiteneffekte, wenn Sie andere Programme rufen.

Aufgabe 2.2: DFA (9 Punkte [2+3+3+1])

Deterministische endliche Automaten wechseln ihren internen Zustand bei jeder als gültig definierten Eingabe in einen eindeutig bestimmten Folgezustand. Nutzen Sie die Datei `dfa.cpp` des Programmrahmens, um den in Abbildung 2 dargestellten Automaten zu simulieren.

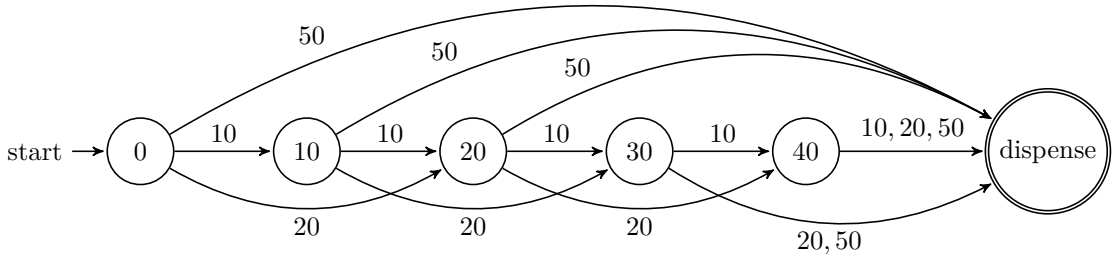

Abbildung 2: Getränkeautomat

An diesem Getränkeautomaten können Flaschen für einen Preis von jeweils 50 Cent erworben werden. Der Automat akzeptiert Münzen im Wert von 10, 20 oder 50 Cent. Sobald insgesamt mindestens 50 Cent eingeworfen wurden, gibt der Automat das Getränk aus und terminiert den Vorgang. Überzähliges Wechselgeld wird nicht zurückgezahlt. Nach Ausgabe eines Getränks wird der Automat in seinen Startzustand zurückgesetzt, ungültige Eingaben werden immer ignoriert.

Die Methode `step` erhält als Parameter den aktuellen Zustand des Automaten (*state*), den Betrag der eingegebenen Münze (*value*) und gibt daraufhin den neuen Zustand zurück. Verwenden Sie dafür keine Berechnungen, sondern ausschließlich `if`- und `switch`-Anweisungen auf den unveränderten Parametern. Der initiale Zustand des Automaten ist „0“ und der Automat erreicht den Zustand „dispense“, sobald das Getränk ausgegeben werden soll.

Hinweis: DFAs haben außer ihrem Zustand keine Möglichkeit zur Speicherung von Informationen. Sie dürfen aus diesem Grund keine Variablen anlegen, in denen Sie bspw. den Preis vorhalten. Der neue Zustand darf ausschließlich aus dem aktuellen Zustand und der Eingabe abgeleitet werden.

a) Vervollständigen Sie die Methode `step`, sodass der oben beschriebene Automat simuliert wird.

b) Der Automat aus der a) soll nun erweitert werden. Der Automat soll zur Auswahl eines Getränkes als erste Eingabe die Zahl eines der folgenden Getränke mit jeweils definiertem Preis erfordern und das Getränk erst ausgeben, wenn die entsprechende Summe erreicht wurde:

   - 1: 50 Cent
   - 2: 80 Cent
   - 3: 100 Cent

   Zeichnen Sie den so erweiterten DFA mit der aus dem Tutorium bekannten Notation.

c) Erweitern Sie die Methode `stepExtended`. Die Funktion soll die in b) definierte erweiterte Funktionalität des DFA abdecken.

   Der Aufruf des erweiterten Modus geschieht im mitgelieferten Programmrahmen, indem bei Programmstart ein beliebiges Kommandozeilenargument übergeben wird (z. B. `dfa.exe 1`).

d) Leiten Sie aus Ihren Ergebnissen aus a-c eine allgemeine Methodik ab, wie Sie einen DFA nur unter Zuhilfenahme eines Zustands und einer Eingabe in Programmcode abbilden können.

Aufgabe 2.3: Reguläre Ausdrücke (6 Punkte)

Reguläre Ausdrücke ermöglichen es, Zeichenketten syntaktisch zu prüfen und zu parsen. In dieser Aufgabe sollen Sie **schriftlich** zu einem Adressinformationsmodell reguläre Ausdrücke ableiten. Eine Adresse bestehe im Folgenden aus diesen Elementen:

- Vorname (*Zeichenkette*)
- Nachname (*Zeichenkette*)
- Straße (*Zeichenkette*)
- Hausnummer (*Nummer ohne führende Null, auch zusammengesetzt oder in Verbindung mit maximal einem Buchstaben*)
- Postleitzahl (*5 Ziffern*)
- Ort (*Zeichenkette*)

Erstellen Sie geeignete reguläre Ausdrücke für die sechs Elemente der Adresse.[1] Achten Sie darauf, dass in Ihren Ausdrücken mindestens die Beispieladressen aus Tabelle 1 komplett auf Korrektheit überprüft werden können und überlegen Sie sich Einschränkungen, die für die Zeichenketten in Deutschland üblicherweise gelten. Erklären Sie für jeden Ausdruck, welche Eingaben gültig sind und geben Sie jeweils ein korrektes und ein fehlerhaftes Beispiel an.

Vorname	**Nachname**	**Straße**	**Nr.**	**PLZ**	**Ort**
Max	Mustermann	Musterstraße	2a	12345	Berlin
Hasso	Plattner	Prof.-Dr.-Helmert-Str.	2-3	14482	Potsdam
Mark-Dieter	Kling	Paul-Heise-Weg	5	15711	Königs Wusterhausen
Bärbel	von Strauß	Am Häuschen	129	74172	Neckarsulm

Tabelle 1: Beispieladressen

[1] Evtl. ist die Ressource `www.regex101.com` für Sie nützlich.

Aufgabe 2.4: Verarbeitung von Textdateien (7 Punkte $^{1+3+1+2}$)

In dieser Aufgabe geht es darum, textuell gespeicherte Informationen strukturiert einzulesen und auszuwerten. Der im Programmrahmen mitgelieferte Datensatz *persons.dat* enthält Informationen zu Personen. Das Modul `fileio.cpp` liest den Datensatz ein, wobei der Dateipfad als Kommandozeilenargument übergeben wird.

a) Implementieren Sie in `readTokensAndLines` das Einlesen der Datei (mittels `std::getline`), wobei jede Zeile elementweise (getrennt durch einen Delimiter) zerlegt wird. Das Programm soll für jeden Eintrag den Namen und die zugehörige IBAN im üblichen 4er-Block-Format auf `std::cout` ausgeben: `[Vorname] [Nachname] - [IBAN]`
Beispiel: `Max Mustermann - DE44 3507 0024 0388 2496 00`

b) Implementieren Sie die Funktion `isValueCorrect`, die einen String-Parameter auf korrekte Syntax überprüft. Es sollen die Spalten *Vorname*, *Nachname* und *IBAN* wie folgt auf Korrektheit überprüft werden:

- Für die Spalte *Vorname* und *Nachname* ist eine Buchstabenfolge aus lateinischen Buchstaben ohne Sonderzeichen und Umlaute zulässig, die mit einem Großbuchstaben beginnt, gefolgt von Kleinbuchstaben, und — im Kontext dieser Aufgabe — mindestens zwei und höchstens 15 Zeichen umfasst.
- Für die Spalte *IBAN* ist eine Zeichenkette beginnend mit zwei Großbuchstaben erlaubt. Es werden nur Großbuchstaben und die Ziffern 0-9 erlaubt. Minimal sind 18 und maximal sind 34 Stellen.

Falls durch das Programm ein syntaktischer Fehler im Eingabedatensatz entdeckt wird, soll eine entsprechende Warnung in eine Datei geschrieben werden. Hierzu sollen die gesamte Zeile des Eingabedatensatzes sowie der jeweilige ermittelte Fehler mittels `ofstream` in eine Logdatei `fileio.log` geschrieben werden. (*Hinweis:* Sie können reguläre Ausdrücke aus der Standardbibliothek `regex` nutzen)

c) Implementieren Sie die Funktion `isCheckDigitCorrect`, die einen String-Parameter (eine IBAN) daraufhin überprüft, ob ihre Prüfziffern korrekt sind. Im Kontext dieser Aufgabe ergeben sich die Prüfziffern (3. und 4. Stelle der originalen IBAN) dadurch, dass zunächst die Großbuchstaben wie folgt ersetzt werden: A=10, B=11, C=12 usw. Anschließend ergeben sich die Prüfziffern, indem die so entstehende Zahl modulo m gerechnet wird[2]. Zur Vereinfachung wird in dieser Aufgabe von m=100 ausgegangen.

d) Beim eigentlichen Verfahren zur Berechnung der Prüfziffern wird m=97 verwendet. Können Sie diese Berechnung ohne Weiteres durchführen? Welches Problem gäbe es? Schreiben Sie Ihre Antwort als Kommentar in `modulo` und implementieren Sie diese Funktion, die zwei Eingabeparameter nimmt und a modulo b zurückgibt.

[2] Ist das Ergebnis einstellig also bspw. 1 so wird mit 0 aufgefüllt also das Ergebnis wäre bspw. 01

Allgemeine Hinweise zur Bearbeitung und Abgabe

- Die Aufgaben können allein oder zu zweit bearbeitet werden. Je Gruppe ist nur eine Abgabe notwendig.
- Bonuspunkte können innerhalb des Übungsblattes verlorene Punkte ausgleichen. Sie sind nicht auf andere Übungsblätter übertragbar und werden nicht über die reguläre Punktzahl hinaus angerechnet.
- Bitte reichen Sie Ihre Lösungen bis spätestens **Dienstag, den 24. Mai um 17:00 Uhr** ein.
- Die Implementierung kann auf einer üblichen Plattform (Windows, Linux, OS X) erfolgen, darf aber keine plattformspezifischen Elemente enthalten, d. h. die Implementierung soll plattformunabhängig entwickelt werden.
- Die Lösungen von theoretischen Aufgaben können als Textdatei oder PDF abgegeben werden. Benennen Sie die Dateien im Format `<Aufgabenblatt>_<Aufgabe>_<Teilaufgabe>`, bspw. „`2_1_d.txt`“.
- Bestehen weitere Fragen und Probleme, kontaktieren Sie den Übungsleiter oder nutzen Sie das Forum im Moodle.
- Archivieren Sie zur Abgabe Ihren bearbeiteten Programmrahmen und die Lösungen der theoretischen Aufgaben als Zip-Archiv und ergänzen Sie Ihre Namen im Bezeichner des Zip-Archivs im folgenden Format: **`uebung2_Grp_MatNr1_MatNr2.zip`**,
  bspw. `uebung2_grp1_123456_654321.zip`. Beachten Sie, dass dabei nur die vollständigen Lösungen sowie eventuelle Zusatzdaten gepackt werden (alle Dateien, die im gegebenen Programmrahmen vorhanden waren); vermeiden Sie bitte nicht notwendige Unterordner. Laden Sie keine Kompilate und temporären Dateien (*.obj, *.pdb, *.ilk, *.ncb, *.exe, etc.) hoch. Testen Sie vor dem Hochladen, ob die Abgabe fehlerfrei kompiliert und ausgeführt werden kann.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
  `https://moodle.hpi.de/course/view.php?id=297`.