technische universität dortmund

fakultät für informatik

Jurij Kuzmic

Dortmund, den 17. Oktober 2019

Übung zur Vorlesung EidP (WS 2019/20)

Blatt 1

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 24. Oktober 2019, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep1920uebung/`

- Die Aufgaben sind in Dreiergruppen zu bearbeiten. Der Lösungsweg und die Programmierung ist gemeinsam zu erarbeiten.
- Die Gruppenmitglieder sollten gemeinsam an der gleichen Übungsgruppe teilnehmen. Die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet. Jedes Gruppenmitglied muss in der Lage sein, alle abgegebenen Lösungen vorzustellen.
- Diese müssen spätestens bis zum jeweiligen Abgabetermin (siehe jeweiliger Aufgabenzettel) abgegeben werden. In darauffolgenden Übungen sind teilweise einzelne abgegebene Lösungen oder auch eine Musterlösung zu präsentieren und zu besprechen.
- Die Abgabe geschieht zwingend über das Web-Interface unter

  `https://ess.cs.tu-dortmund.de/ASSESS/`

  Eine Abgabe per Email wird nicht akzeptiert. Dieses Programm fragt nach Informationen über die Gruppenteilnehmer (**Vor-, Nachnamen, Matrikelnummern**) und sammelt im aktuellen Verzeichnis die abzugebenden Dateien ein. Namen und Anzahl von abzugebenden C++-Quellcodedateien variieren und stehen in der jeweiligen Aufgabenstellung. Wählen sie dazu **Abgabe** und als Veranstaltung **EidP Übungen WS 2019/20** aus und laden Sie Ihre Ergebnisdateien hoch. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden. Bewertet wird immer die **letzte** von Ihnen hochgeladene Version.
- Die abgegebenen Antworten/Programme werden automatisch auf Ähnlichkeit mit anderen Abgaben überprüft. Werden hierbei starke Übereinstimmungen festgestellt, wird die Aufgabe als nicht abgegeben bewertet. **Plagiieren kann zum Nichtbestehen der Übung führen.**

- Sobald eine Abgabe von den Betreuern korrigiert wurde, können die erzielte Punktzahl und die korrigierte Lösung ebenfalls unter dieser Adresse eingesehen werden.
- Es ist ratsam, die Programme vor der Abgabe zu kompilieren und auszuführen.
- Verwenden Sie für die Textaufgaben **reine Textdateien.** Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!

Aufgaben

Aufgabe 1: Grundlagen (0.8 Punkte)

Legen Sie für Ihre Antworten eine Text-Datei `Aufgabe_01_1.txt` an.

a) Geben Sie den Unterschied zwischen den *einfachen* und den *zusammengesetzten Datentypen* an. (0.1 Punkte)

b) Was ist eine *Datendefinition*? Was geschieht dabei? (0.2 Punkte)

c) Welche der folgenden Zeichenketten (getrennt durch Komma) können als ein *Bezeichner* benutzt werden:

- `0011`
- `%f?`
- `auto`
- `this`
- `_bz_`
- `ja_ö//`

Für jede Zeichenkette, die kein Bezeichner sein kann, erklären Sie, warum nicht.
(0.3 Punkte)

d) Erklären Sie am Beispiel des Dividenden 123 und des Divisors 25 die Funktionsweise der Operatoren `%` und `/`. (0.2 Punkte)

Aufgabe 2: Zahlendarstellung (1.2 Punkte)

Legen Sie für ihre Lösungen die Datei `Aufgabe_01_2.txt` an.

a) Wandeln Sie folgende Zahlen zwischen Dezimal- und Binärdarstellung um. Geben Sie jeweils den Rechenweg an. Wandeln Sie

- 15754 aus Dezimal- in Binärdarstellung für Ganzzahlen ohne Vorzeichen,
- 110101011110011 aus (unsigned) Binär- in Dezimaldarstellung, und
- -1338 aus Dezimal- in 12-Bit-Repräsentation um.

(0.4 Punkte)

b) Gegeben Sei folgender Algorithmus:

1. Initialisiere $i = 31, n = 1$
2. Solange $i \geq 1$ gilt:
3. Setze $i = i - 1$
4. Setze $n = n \cdot 2$
5. Gib n aus

Welchen Wert gibt der Algorithmus aus, falls n und i 32-Bit Ganzzahlen

1. ohne Vorzeichen, oder
2. mit Vorzeichen sind.

Begründen Sie Ihre Antworten. (0.4 Punkte)

c) Was gilt für eine Zahl in Binärdarstellung, deren zwei niedrigstwertigen Bits 0 sind? Was gilt für eine Zahl in Binärdarstellung, deren $k \in \mathbb{N}$ niedrigstwertigen Bits 0 sind? (0.4 Punkte)

Aufgabe 3: Gewichtete alternierende Summe (2 Punkte)

Sei $A = (a_1, \ldots, a_n)$ eine Folge von reellen Zahlen. Mit A_s definieren wir die *gewichtete alternierende Summe* dieser Zahlen, wobei a_i mit $\frac{1}{i}$ gewichtet werden soll:

$$A_s := \sum_{i=1}^{n} \frac{a_i}{i} \cdot (-1)^{(i-1)} = \frac{a_1}{1} - \frac{a_2}{2} + \frac{a_3}{3} - \ldots \pm \frac{a_n}{n}$$

Beispiel: Mit $A = (5, 9, 3, 7)$ ist die gewichtete alternierende Summe $A_s = -0,25$.

Legen Sie zunächst eine Datei `Aufgabe_01_3.cpp` an. Ergänzen Sie darin das folgende C++-Programmfragment so, dass bei Ausführung des Programms der Variable `altsum` der Wert der gewichteten Summe zugewiesen wird. Die gewichtete alternierende Summe soll auch für beliebige andere Zahlen im Array `A` korrekt berechnet werden. An dieser Stelle wird eine Schleife benötigt.

Hinweis: Nutzen Sie für die Bestimmung der Vorzeichen der Summanden den Operator `%`.

```
1  /*** Aufgabe_01_3.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    double A[] = { 8.128, 101.02, 84.89, 58.95, 5.6,
7    53.58, 45.3, 79.01, 23.53, 45.18 };
8
9    int n = 10; // Anzahl Elemente in A
10   double altsum = 0;
11
12   //************************************
13   // Hier beliebig viele Zeilen ergaenzen
14   //************************************
```

```
15
16    cout << "Gew. alt. Summe: " << altsum << endl;
17    return 0;
18  }
19  /* Ausgabe:
20  Gew. alt. Summe: 0
21  */
22  /*** Ende Aufgabe_01_3.cpp ***/
```

Kompilieren Sie das Programm und führen Sie es anschließend aus. Kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der Datei `Aufgabe_01_3.cpp`.

Präsenzaufgabe 4: Grammatik (0 Punkte)

Sei $G = (N, T, S, P)$ eine kontextfreie Grammatik, mit

- $N = \{S, A, B, C\}$
- $T = \{0, 1, +, -, (,)\}$
- und P:

$$\begin{aligned} S &\rightarrow A \mid A + A \\ A &\rightarrow (A) \mid B - B \\ B &\rightarrow B + B \mid A \mid C \\ C &\rightarrow 0 \mid 1 \end{aligned}$$

1. Erläutern Sie kurz, in welcher Beziehung Grammatiken und Programme im Zusammenhang von Programmiersprachen zueinander stehen.
2. Beschreiben Sie die einzelnen Bestandteile von G.
3. Leiten Sie mit G drei unterschiedliche Wörter ab und geben Sie jeweils den Ableitungsbaum an.
4. Können Sie $((0 + 0 + 1 + 1))$ ableiten? Begründen Sie Ihre Antwort.
5. Wie lang ist das längste Wort, das man mit dieser Grammatik erzeugen kann?