technische universität dortmund

fakultät für informatik

Jurij Kuzmic
Horst Schirmeier
Jonas Ellert

Dortmund, den 12. November 2020

Übung zur Vorlesung EidP (WS 2020/21)

Blatt 1

Block gelb

Es können 4 Punkte erreicht werden.

Abgabedatum: 19. November 2020, 23:59 Uhr

Hinweise

- Bitte beachten Sie aktuelle Hinweise unter:

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep2021uebung/`

- Wir empfehlen zum Bearbeiten der Programmieraufgaben die Entwicklungsumgebung *Visual Studio Code*. Für die **Installation auf Ihrem Rechner** stellen wir auf der Übungs-Webseite eine detaillierte Installationsanleitung bereit.
- Die Aufgaben sind **in Dreiergruppen** zu bearbeiten, die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet. Der Lösungsweg und die Programmierung sind gemeinsam zu erarbeiten. **Ausnahmsweise sind bei Blatt 1 auch Einzelabgaben erlaubt.**
- Der Abgabetermin für jedes Aufgabenblatt ist jeweils oben vermerkt. Die Abgabe geschieht über das ASSESS-System: `https://ess.cs.tu-dortmund.de/ASSESS/`
  Dieses benötigt Informationen über die Gruppenteilnehmer (**Vor-, Nachnamen, Matrikelnummern**) und fordert Sie auf, die zur Aufgabe gehörenden C++-Quelltextdateien und ggf. weitere Dateien (siehe Aufgabenstellung) hochzuladen. Wählen Sie dazu **Abgabe** und als Veranstaltung **Einführung in die Programmierung (EidP) Übung** aus, und laden Sie Ihre Dateien hoch. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden, bewertet wird immer die **letzte** von Ihnen hochgeladene Version.
- Die abgegebenen Dateien 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 im ASSESS-System eingesehen werden.
- Sie sollten während der Entwicklung Ihrer Programme diese unbedingt regelmäßig – und insbesondere noch einmal vor der Abgabe – **compilieren und ausführen**. Ein solches iteratives Vorgehen ist grundlegend für jede Programmiertätigkeit!
- 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 Textdatei `Aufgabe_01_1.txt` an.

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

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

c) Welche der folgenden Zeichenketten können als *Bezeichner* benutzt werden?

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

Begründen Sie die Fälle, die *keine* Bezeichner sein können. (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 in der letzten Zeile 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 niederwertigste Bits 0 sind? Was gilt für eine Zahl in Binärdarstellung, deren $k \in \mathbb{N}$ niederwertigste 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 in Ihrer C++-Entwicklungsumgebung (siehe Abschnitt **Hinweise** auf der ersten Seite) 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. Implementieren Sie die Summe für das vorgegebene 5-elementige Array. An dieser Stelle *können* Sie eine Schleife verwenden, damit das Programm auch für den Fall $n \neq 5$ funktioniert.

Hinweis: Falls Sie in Ihrer Implementierung eine Schleife verwenden, können Sie zum Bestimmen der Vorzeichen der Summanden den Operator `%` nutzen.

```
1  /*** Aufgabe_01_3.cpp ***/
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    double A[] = { 3.2, 200.3, 83.22, 4.68, 564.0 };
7
8    int n = 5; // Anzahl Elemente in A
9    double altsum = 0;
10
11   //************************************
12   // Hier beliebig viele Zeilen ergaenzen
13   //************************************
14
15   cout << "Ergebnis: " << altsum << endl;
16   return 0;
17 }
18 /* Ausgabe:
19 Ergebnis: 0
20 */
21 /*** Ende Aufgabe_01_3.cpp ***/
```

Kompilieren Sie das Programm und führen Sie es anschließend aus; falls Sie die Entwicklungsumgebung *Visual Studio Code* nutzen, finden Sie in der Installationsanleitung auf der Übungs-Webseite eine Beschreibung der dafür notwendigen Schritte. Kopieren Sie die Ausgabe des Programms als Block-Kommentar an das Ende der Datei `Aufgabe_01_3.cpp`.

Präsenzaufgabe 4: Grammatik (0 Punkte)

Hinweis: Präsenzaufgaben werden **nicht abgegeben und bewertet**, sondern gemeinsam mit dem Übungsleiter während der Übung erarbeitet, in der auch der Rest des vorliegenden Übungsblattes 1 besprochen wird. Sehen Sie sich bitte vorher die entsprechenden Vorlesungsinhalte (hier: den Exkurs zu *Grammatiken* in Kapitel 2 der Vorlesung) noch einmal an.

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?