Abteilung für Angewandte Mathematik
Prof. Dr. S. Bartels
M.Sc. C. Palus

18.06.2018

Übung zur Vorlesung

Einführung in die Programmierung

SoSe 2018 – Blatt 9

Abgabe: Briefkästen RZ/E-Mail bis Montag, den 25.06.2018 um 16:00 Uhr

Aufgabe 1 (10 Punkte)**.** Erstellen Sie eine Liste aller syntaktischen und semantischen Fehler im folgenden Quelltext eines C++-Programms, das den größten gemeinsamen Teiler zweier eingegebener Zahlen `zahl1` und `zahl2` bestimmen sollte. Geben Sie jeweils unter Angabe der Zeilennummer an, wie der Quelltext in dieser Zeile korrekterweise lauten müsste.
Hinweis: Es sind zehn Fehler zu finden.

```
1  /*Programm zur Berechnung des ggT zweier Zahlen
2  #include iostream
3
4  bool zahleingabe( int num ) {
5    int eingabe = 0;
6    std::cout >> "Bitte geben Sie die " >> nummer >> ". Zahl ein: ";
7    std::cin << eingabe;
8    return eingabe;
9  }
10
11 int ggt( int a, int b ) {
12   if ( a=0 ) {
13     return 0;
14   }
15   else {
16     while ( b!=0 ) {
17       if ( a>b ) {
18         a-=b;
19       }
20       else {
21         b-=a;
22       }
23     }
24     return a;
25 }
26
27 void loesungsausgabe(int zahl1, int zahl2) {
28   std::cout << "Der ggT von " << zahl1 << " und " << zahl2;
29   std::cout << " ist " << ggt( zahl1, zahl2 ) << "!\n"
30 }
31
32 int main() {
33   int zahl1 = zahleingabe( 1 );
34   int zahl2 = zahleingabe( 2 );
35   loesungsausgabe( zahl1, zahl2 );
36   return 0;
37 }
```

Aufgaben 2, 3 und 4 auf der Rückseite

Aufgabe 2 (10 Punkte)**.** Die Beobachtung, dass sich die Anzahl an Transistoren, die in einen integrierten Schaltkreis fester Größe passen, alle 24 Monate verdoppelt, wird häufig als das *mooresche Gesetz* bezeichnet. Es ist benannt nach dem Ingenieur Gordon E. Moore, der diese Beobachtung 1965 veröffentlichte. Vereinfachend kann man annehmen, dass eine ähnliche Gesetzmäßigkeit auch für die in einer begrenzten Zeit $T > 0$ verfügbare Rechenkapazität besteht. Um größer werdende Probleme zu lösen, ist es jedoch lohnenswert, sich nicht allein auf hardwareseitige Verbesserungen zu verlassen. Als Beispiel sei ein numerisches Verfahren zur Lösung eines linearen Optimierungsproblems genannt, dessen Lösung im Jahr 1990 noch 10 Tage dauerte. Dank schnellerer Rechner hätte man die Lösung im Jahr 2014 nach Moores Gesetz in etwas mehr als 2 Minuten berechnen können, was gerade eine Verbesserung um den Faktor $2^{24/2} \approx 6.500$ darstellt. Durch Verbesserungen in der Implementierung des Verfahrens konnte man das Problem jedoch im selben Jahr bereits in einer Rechenzeit von ca. 1 Sekunde lösen, was einem Verbesserungsfaktor um 870.000 entspricht!
Erläutern Sie, wie sich die Maximalgröße des Problems, welches in der Zeit T von einem Algorithmus gelöst werden kann, ändert, wenn dieser in Abhängigkeit von der Problemgröße n den folgenden Aufwand besitzt:

$$\text{(i) } c_1 n, \quad \text{(ii) } c_2 n^p \text{ mit } p > 1, \quad \text{(iii) } c_3 \log(n), \quad \text{(iv) } c_4 e^n.$$

Begründen Sie, warum es daher erstrebenswert ist, zur Lösung größer werdender Probleme einen Algorithmus mit polynomiellem Aufwand zu entwickeln, anstatt bei Benutzung eines Algorithmus mit exponentiellem Aufwand die Rechenleistung immer weiter zu erhöhen.

Aufgabe 3 (5+5 Punkte)**.** (i) Besuchen Sie die Seite

`https://www-m11.ma.tum.de/karpfing/matlab-online-kurs/lektion-1/`

und sehen Sie sich dort das Einführungsvideo (6:20 min) zu MATLAB an. Beantworten Sie anschließend die folgenden Fragen:

Mit welchem Befehl berechnet man in MATLAB die dritte Potenz der Summe von π und dem Produkt der Variablen `a` und `b`?
Was ist die Funktion der Variable `ans`?
Was bewirkt ein Semikolon hinter einer Anweisung?
Was bewirkt der Befehl `clc`?
Wie kann man sich die Kurzanleitung zu einem MATLAB-Befehl anzeigen lassen?

(ii) Verwenden Sie den Befehl `plot()`, um die Graphen der folgenden Funktionen zu zeichnen:

$$f_1(x) = \log(x), \quad f_2(x) = x, \quad f_3(x) = x^2, \quad f_4(x) = e^x.$$

Dabei sollen alle Graphen im selben Koordinatensystem dargestellt werden. Benutzen Sie die Befehle `xlabel()` und `ylabel()`, um die Koordinatenachsen zu beschriften, den Befehl `xlim()`, um den dargestellten Bereich der x-Achse auf einen sinnvollen Wert zu ändern, sowie den Befehl `legend()`, um eine Legende der Funktionsgraphen zu erstellen. Geben Sie einen Ausdruck des Plots sowie der von Ihnen verwendeten Befehle ab.

Aufgabe 4 (10 Punkte)**.** Erweitern Sie ihr Sudokuprogramm vom letzten Blatt um eine rekursive Funktion `bool solve( int grid[9][9] )`, welche ein teilweise ausgefülltes 9×9-Gitter unter Einhaltung der Sudokuregeln rekursiv vervollständigt. Übersetzen Sie dazu den Algorithmus aus Aufgabe 2, Blatt 8, in die Sprache C++. Leere Kästchen können Sie durch den Eintrag 0 kennzeichnen. Auf der Vorlesungshomepage finden Sie eine Datei `sudoku.cc`, in welcher ein Grundgerüst vorgegeben ist, das Sie zur Lösung dieser Aufgabe verwenden können. Ebenso finden Sie auf der Vorlesungshomepage eine Headerdatei `sudoku.hh`. Diese können Sie herunterladen und mit dem Befehl `#include "sudoku.hh"` in Ihr Programm einbinden. Danach können Sie die Funktion `draw_sudoku()` mit einem Integer-Array der Größe 9×9 als Parameter aufrufen, um eine übersichtliche Ausgabe eines Sudokugitters in der Konsole zu erhalten (wobei die Funktion `draw_sudoku()` den Eintrag 0 als leeres Kästchen interpretiert). Testen Sie Ihr Programm anhand des Beispielsudokus von Blatt 5.

Abgabe der Programme per E-Mail, (handschriftlich) kommentierte Ausdrucke der Programme und Rechnungen auf gehefteten, mit Namen versehenen Zetteln in die Briefkästen

Homepage zur Vorlesung: `https://aam.uni-freiburg.de/agba/lehre/ss18/einfprog`