Prüfung — Informatik D-MATH/D-PHYS
23. 01. 2008 09:00–11:00

Dr. Bernd Gärtner, Prof. Juraj Hromkovič

Kandidat/in:

Name: .....

Vorname: .....

Stud.-Nr.: .....

Ich bezeuge mit meiner Unterschrift, dass ich die Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden allgemeinen Bemerkungen gelesen und verstanden habe.

Unterschrift: .....

Allgemeine Bemerkungen und Hinweise:

1. Überprüfen Sie die Vollständigkeit der ausgeteilten Prüfungsunterlagen (1 einseitig bedrucktes Deckblatt und 2 zweiseitig bedruckte Aufgabenblätter mit insgesamt 6 Aufgaben)!
2. Falls Sie während der Prüfung durch irgendeine Behinderung oder Störung beeinträchtigt werden, melden Sie dies bitte sofort der Aufsichtsperson! Spätere Klagen können nicht akzeptiert werden.
3. Erlaubte Hilfsmittel: **keine**.
4. Betrugsversuche führen zu sofortigem Ausschluss und können rechtliche Folgen haben.
5. Pro Aufgabe ist höchstens eine gültige Version eines Lösungsversuchs zulässig. Streichen Sie ungültige Lösungsversuche klar durch! Schreiben Sie auf separate Blätter, nicht auf die Aufgabenblätter!
6. Sie dürfen die Aufgaben in beliebiger Reihenfolge lösen. Konzentrieren Sie sich jeweils auf eine Aufgabe, aber teilen Sie sich Ihre Zeit ein!
7. Nach Ablauf der Prüfungszeit verlassen Sie bitte den Raum und lassen Sie nur die Blätter auf Ihrem Platz liegen, die zur Abgabe bestimmt sind! **Diese müssen alle mit Ihrem Namen beschriftet sein. Die Prüfungsblätter sind dabei mit abzugeben!**
8. Die Prüfung ist bestanden, wenn Sie 60 von 120 erreichbaren Punkten erzielen.

1	2	3	4	5	6	$\sum$

Aufgabe 1. (18 Punkte) Werten Sie die folgenden Ausdrücke von Hand aus und geben Sie dabei alle Zwischenschritte an. Sie können annehmen, dass `x` eine Variable vom Typ `int` mit dem Wert 1 ist.

a) `1 + 7 / 2`

b) `x == 1 || 1 / (x - 1) < 1`

c) `!(1 && x) + 1`

Aufgabe 2 (20 Punkte) Während Ihres Praktikums bei der Softwarefirma Ugly Solutions Ltd. zeigt Ihnen Ihre Chefin in einem (unkommentierten) Programm folgende Funktion, die offenbar testet, ob eine gegebene natürliche Zahl `n` "hässlich" ist.

```
bool haesslich (unsigned int n)
{
  return
    (n == 1) ||
    (n % 2 == 0) && haesslich (n / 2) ||
    (n % 3 == 0) && haesslich (n / 3) ||
    (n % 5 == 0) && haesslich (n / 5);
}
```

a) Ihre Chefin hat das Gefühl, die Ursache für rätselhafte Abstürze des Gesamtprogramms sei möglicherweise in der Funktion `haesslich` zu finden. Gibt es in der Tat ein Problem mit dieser Funktion, und wenn ja, welches? Wie könnten Sie es beheben?

b) Ihre Chefin möchte das Programm besser verstehen und deshalb von Ihnen wissen, was eigentlich eine hässliche Zahl ist. Geben Sie Ihr eine Definition, die genau zu der (von Ihnen "reparierten") Funktion passt!

Aufgabe 3. (20 Punkte) Zwei natürliche Zahlen sind teilerfremd, wenn sie keinen gemeinsamen Teiler grösser als 1 haben. Zum Beispiel sind 21 und 10 teilerfremd, weil $\{3,7\} \cap \{2,5\} = \emptyset$. Hingegen sind 21 und 15 nicht teilerfremd, weil $\{3,7\} \cap \{3,5\} \neq \emptyset$. Implementieren Sie eine Funktion, die auf Teilerfremdheit prüft.

```
// PRE: a > 0, b > 0
// POST: Der Rueckgabewert ist true g.d.w. a und b teilerfremd sind.
bool teilerfremd(unsigned int a, unsigned int b);
```

Tipp: Sie können die Aufgabe zum Beispiel lösen, indem Sie eine Hilfsfunktion implementieren, die Sie schon aus der Vorlesung kennen. Bitte beachten Sie jedoch, dass Sie diese Implementation auch tatsächlich hinschreiben müssen. Es reicht hier nicht, einfach auf die Unterlagen zu verweisen.

Aufgabe 4 (18 Punkte) Gehen Sie von folgenden Fakten und Definitionen aus:

1. Ein Algorithmus ist eine Handlungsvorschrift zur Entscheidung eines bestimmten Problems. Die Eingabe muss endlich sein, zum Beispiel eine Zahl $n \in \mathbb{N}$ oder aber auch ein Programmtext $P \in \mathcal{P}$ aus der Menge aller Programme. Die Ausgabe ist entweder JA oder NEIN. Ein Algorithmus kommt in jedem Fall nach einer endlichen Anzahl Schritten zur Ausgabe.

2. Ein Programm ist eine Handlungsvorschrift, realisiert in einer bestimmten Programmiersprache (zum Beispiel C++). Der entscheidende Unterschied zu einem Algorithmus ist, dass ein Programm auf bestimmten Eingaben auch unendlich lange arbeiten kann. Wir sagen, dass ein Programm eine Eingabe akzeptiert, wenn die Ausgabe JA ist, und dass es eine Eingabe nicht akzeptiert, wenn die Ausgabe NEIN ist oder wenn das Programm unendlich lange arbeitet.

3. Es gibt keinen Algorithmus UNIV, der für ein beliebiges Programm $P \in \mathcal{P}$ und eine beliebige Eingabe e entscheidet, ob P die Eingabe e akzeptiert oder nicht.

Sei HALT ein Algorithmus, der für ein beliebiges Programm $P \in \mathcal{P}$ und eine beliebige Eingabe e entscheidet, ob P auf der Eingabe e hält oder nicht. Ihre Aufgabe ist es zu zeigen, dass es keinen Algorithmus HALT geben kann. Dazu müssen Sie argumentieren, dass man UNIV mit Hilfe von HALT realisieren könnte. Dadurch führen Sie die Annahme, dass es HALT gibt, zu einem Widerspruch.

Aufgabe 5 (20 Punkte) Betrachten Sie das folgende Programm zur Berechnung der Zahl 2 beruhend auf der Formel

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 2.$$

Das Programm berechnet dabei zunächst die Summe

$$\sum_{i=0}^{999} \left(\frac{1}{2}\right)^i$$

in einer Variablen `two` und gibt dann deren Differenz zum gewünschten Ergebnis 2 aus.

```
#include<iostream>

int main()
{
  float two = 1; // (1/2)^0
  float p = 1;   // (1/2)^0
  for (int i=1; i<1000; ++i) {
    p /= 2;      // (1/2)^i
    two += p;
  }

  std::cout << two - 2 << "\n";
  return 0;
}
```

a) Schreiben Sie hin, was der Wert der Variablen `two` nach den ersten 4 Schleifendurchgängen ist, und beschreiben Sie kurz wie der Verlauf in den folgenden Durchgängen sein wird. Benutzen Sie dazu die Darstellung $\pm d_0.d_1d_2d_3\ldots$, wobei $d_i \in \{0, 1\}$.

b) Was ist die Ausgabe des Programms? Wählen Sie eine der folgenden drei Antworten aus und begründen Sie diese, wobei wir annehmen, dass der Zahlentyp `float` dem Fliesskommastandard IEEE 754 folgt. Für diese Aufgabe ist das Rundungsverhalten der Fliesskommazahlen relevant. Laut Standard geht das wie folgt: Wenn das exakte Resultat nicht als Fliesskommazahl darstellbar ist, dann wird zur nächsten darstellbaren Zahl gerundet. Wenn das exakte Resultat genau zwischen zwei Fliesskommazahlen liegt, dann wird zu derjenigen gerundet, die im kleinsten signifikanten bit eine 0 hat (sogenanntes round-to-even).

   i) die Ausgabe ist eine sehr kleine negative Zahl.

   ii) die Ausgabe ist eine sehr kleine positive Zahl.

   iii) die Ausgabe ist exakt 0.

Aufgabe 6 (24 Punkte) Das Ziel dieser Aufgabe ist es, Teile des Strategiespiels Tic Tac Toe als C++ Klasse zu realisieren. Für diejenigen, die das Spiel nicht kennen, hier die Regeln kurz zusammengefasst. Das Spiel wird auf einem anfänglich leeren Brett mit 3×3 Feldern gespielt. Die beiden Spieler markieren abwechslungweise eines der leeren Felder mit ihrem Symbol (`x` und `o`). Gewonnen hat derjenige Spieler, der als erster drei seiner Symbole in einer geraden Linie auf dem Spielfeld stehen hat (d.h. senkrecht, waagerecht, oder diagonal). Wenn alle Felder belegt sind, jedoch keiner der Spieler eine Gewinnstellung erreicht hat, dann endet das Spiel unentschieden.

a) Deklarieren Sie die Klasse `TicTacToe` und deren Datenmitglieder. Legen Sie eine Semantik fest, welche Werte der Datenmitglieder was bedeuten. Denken Sie daran, dass die Klasse in der Lage sein muss, jede gültige Spielposition zu kodieren.

b) Implementieren Sie einen Konstruktor `TicTacToe()` für die Klasse, der alle von Ihnen deklarierten Datenmitglieder korrekt initialisiert (leeres Spielfeld).

c) Implementieren Sie die folgende Mitgliedsfunktion, die überprüft, wer das Spiel gewonnen hat. Vervollständigen Sie dabei auch die Postcondition derart, dass daraus klar hervorgeht, was der Wert der Rückgabe bedeutet. Beachten Sie, dass sie nicht überprüfen müssen, ob das Spielbrett gegebenenfalls schon voll ist. Der Fall "es kann keine Aussage getroffen werden" kann also zweierlei bedeuten, nämlich dass das Spiel noch läuft oder dass ein Unentschieden vorliegt.

```
// PRE: Das Spiel befindet sich in einem gueltigen Zustand.
// POST: Die Rueckgabe ist __ falls Spieler 1 gewonnen hat,
//                         __ falls Spieler 2 gewonnen hat und
//                         __ falls keine Aussage getroffen werden kann.
int gewonnen(){...}
```

Tipp: Felder (engl. Arrays) müssen nicht, dürfen aber verwendet werden.