Prüfung — Informatik D-MATH/D-PHYS
10. 8. 2011 14:00–16:00

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 allgemeinen Bemerkungen gelesen und verstanden habe.

Unterschrift: .....

Allgemeine Bemerkungen und Hinweise:

1. Überprüfen Sie die Vollständigkeit der ausgeteilten Prüfungsunterlagen (drei Blätter, bestehend aus 1 Deckblatt und 2 Aufgabenblättern 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. Einzige Ausnahme sind Wörterbücher**.
4. Betrugsversuche führen zu sofortigem Ausschluss und können rechtliche Folgen haben.
5. Pro Aufgabe ist höchstens eine gültige Version eines Lösungsversuches 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 oder wenn Sie frühzeitig abgeben möchten, übergeben Sie Ihre Lösungen bitte einer Aufsichtsperson und verlassen zügig den Raum. **Jedes Ihrer Lösungsblätter muss mit Ihrem Namen beschriftet sein. Die Prüfungsblätter sind mit abzugeben!**
8. Die Prüfung ist bestanden, wenn Sie 60 von 120 erreichbaren Punkten erzielen.

1	2	3	4	5	6	$\sum$

Aufgabe 1. (10 Punkte) Geben Sie zu den folgenden Ausdrücken jeweils den Typ und den Wert an. Es gibt keine Punkte für die Teilschritte der Auswertung. Sie brauchen diese nicht hinzuschreiben. Gehen Sie zu Beginn *jeder* der Teilaufgaben von folgenden Deklarationen aus:

```
int k = 1;
int z = 1;
double d = 1.5;
float f = 0.25f;
```

a) `false && 1.77 / d == 1.18`

b) `127 % 7 * 67 % 17 % k`

c) `&k == &z`

d) `9 / 4.0f + 1 - f * 5`

e) `--z + k++ + d`

Aufgabe 2. (25 Punkte) Diese Aufgabe besteht aus fünf unabhängigen und selbsterklärenden Teilaufgaben, von denen jede fünf Punkte gibt.

a) Was ist der Unterschied für den Anwender zwischen den beiden folgenden Deklarationen? (Ein Satz reicht als Antwort)

```
class foo {
  int i;
};
```

```
struct foo {
  int i;
};
```

b) Wie oft wird der Rumpf der folgenden Schleife ausgeführt? Sie müssen keine Begründung angeben.

```
int i = 3.7;
while (i > 0) {
  i = i - 0.6;
}
```

c) Was ist die Ausgabe des folgenden Codes?

```
for (int i = 1; i % 5 != 0; ++i) {
  if (i % 4 == 0) ++i;
  std::cout << i << " ";
}
```

d) Wie lautet die binäre Fliesskommadarstellung der Dezimalzahl 1.625?

e) Sei p/q eine rationale Zahl. Ist die folgende Aussage wahr oder falsch: Es gibt eine Basis, so dass p/q in einem Fliesskommasystem zu dieser Basis endlich, d.h. *ohne Periode*, darstellbar ist. Geben Sie eine kurze Begründung an.

Aufgabe 3. (25 Punkte) Betrachten Sie folgendes Registermaschinen-Programm. Geben sie eine ausführliche Vor- und Nachbedingungen an. Das heisst, beschreiben Sie, wie die Ausgabe des Programmes ist, in Abhängigkeit der Eingabe (Nachbedingung). Beschreiben Sie auch, wie die Eingabe gestaltet sein muss, damit das Programm richtig funktioniert (Vorbedingung). Die Vorbedingung muss insbesondere sicherstellen, dass das Programm keine Endlosschleife erzeugt.

```
 1. Register(1) ⟵ 1
 2. Read into Register(2)
 3. If Register(2) = 0, then go to line 15
 4. Read into Register(3)
 5. Register(2) ⟵ Register(2) - Register(1)
 6. If Register(2) = 0, then go to line 15
 7. Read into Register(4)
 8. If Register(3) <= Register(4), then go to line 10
 9. Go to line 13
10. Register(2) ⟵ Register(2) - Register(1)
11. Register(3) ⟵ Register(4)
12. Go to line 6
13. Output ⟵ "No"
14. Go to line 16
15. Output ⟵ "Yes"
16. End
```

Aufgabe 4. (10 / 20 Punkte) In dieser Aufgabe sollen sie eine Funktion programmieren, die das nächste Glied der sogenannten Morris-Sequenz berechnet. Um von einer Eingabezahl den Morris-Nachfolger zu berechnen, "liest" man die Ziffern der Zahl von links nach rechts und gruppiert dabei mehrfach vorkommende Ziffern. Die Zahl 112 z.B. kann man als "zwei Einsen und eine Zwei" lesen. Übersetzt man das in Ziffern, so entsteht die Zahl 2112. Diese wiederum kann man lesen als "eine Zwei, zwei Einsen und eine Zwei". Demnach ist das nächste Glied 122112, und so weiter. Hier der Beginn der Sequenz, wenn man mit 1 anfängt.

$$1, 11, 21, 1211, 111221, 312211, 13112221, \ldots$$

Hinweis: Wie sie sehen, kommt in der Sequenz oben keine Ziffer grösser als 3 vor. Davon können Sie für die ganze Aufgabe ausgehen. (Wenn die Eingabezahl weder Ziffern grösser als 3 noch Folgen von mehr als 3 gleichen Ziffern enthält, ist dies relativ einfach zu beweisen.)

```
// PRE: [b,e) ist ein nicht-leerer, gueltiger Bereich,
//      in dem die einzelnen Ziffern der Eingabezahl
//      abgelegt sind. Das heisst, b zeigt auf die erste
//      Ziffer im Bereich, und e ist der sogenannte
//      past-the-end Zeiger.
//      res ist ein Zeiger auf einen Bereich, in dem die
//      Ziffern der Ausgabezahl abgelegt werden koennen.
//      Dieser Bereich hat ausreichende Laenge.
// POST: In dem Bereich beginnend beim Zeiger res sind die
//       Ziffern der Ausgabezahl abgelegt. In n ist die
//       Anzahl der Stellen der Ausgabezahl gespeichert.
void morris(const int* b, const int* e, int* res, unsigned int& n);
```

a) Beschreiben Sie in Worten wie Ihr Algorithmus funktioniert.

b) Implementieren Sie die Funktion, deren Signatur oben angegeben ist.

Aufgabe 5. (5 / 5 Punkte) Betrachten Sie die folgende Funktion und beantworten Sie die Fragen.

```
unsigned int f(unsigned int n) {
  if (n <= 1) return n;
  else return f(n-1) + n;
}
```

a) Was berechnet die Funktion `f`?

b) Geben Sie eine Methode an, mit der das Gewünschte schneller berechnet werden kann.

Aufgabe 6. (10 / 10 Punkte) Die Bewohner des Planeten Plato sind ein streng hierarchisch organisiertes Volk, und so kommt es, dass jeder Bewohner einen sogenannten Meister hat, dem er regelmässig Rechenschaft ablegen muss. Jeder Bewohner hat genau einen Meister, ein Meister aber kann mehrere Bewohner betreuen. Ein Meister ist selbst auch ein Bewohner und hat somit seinerseits auch einen Meister. Die einzige Ausnahme sind die Könige. Sie betreuen andere Bewohner, haben aber keinen Meister. Es gibt mehrere Könige.

Das platonische Amt für Soziales hat eine Studie in Auftrag gegeben, bei der die Altersstruktur der Gesellschaft im Hinblick auf die Meister-Beziehungen untersucht werden soll. Dazu wird eigens eine Software entwickelt. Ihre Aufgabe ist es, dabei behilflich zu sein. Die Bewohner sollen in einem Array gespeichert werden, so dass jeder Bewohner einen eindeutigen Index hat. Bearbeiten Sie die beiden folgenden Teilaufgaben.

a) Deklarieren Sie ein `struct Bewohner`, in welchem die folgenden Informationen gespeichert werden können:

   1. Initialen des Bewohners (Anfangsbuchstaben des Vor- und Nachnamens)
   2. Alter des Bewohners in Jahren
   3. Der Index des Meisters des Bewohners (-1 falls der Bewohner ein König ist)

b) Nehmen Sie an, dass Sie den Index eines beliebigen Bewohners bekommen, und herausfinden wollen, wer sein König ist. Implementieren Sie eine Funktion, die dies tut.

```
// In diesem globalen Array sind alle Bewohner gespeichert und
// korrekt initialisiert, wobei anz_bew eine zuvor initialisierte
// Konstante ist.
Bewohner bewohner[anz_bew];

// PRE: Die Meister-Struktur von Plato enthaelt keine Zyklen.
//      index ist ein gueltiger Index, 0 <= index < anz_bew.
// POST: Der Rueckgabewert ist der Index des Koenigs von
//       bewohner[index], oder -1 falls bewohner[index] selbst
//       ein Koenig ist.
int finde_koenig(int index);
```

Hinweis: Wenn die Meister-Struktur keine Zyklen enthält, bedeutet das, dass z.B. "M ist Meister von N; N ist Meister von O; O ist Meister von M." *nicht* vorkommen kann.