Prüfung — Informatik D-MATH/D-PHYS
26. 01. 2012 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 allgemeinen Bemerkungen gelesen und verstanden habe.

Unterschrift: .....

Allgemeine Bemerkungen und Hinweise:

1. Überprüfen Sie die Vollständigkeit der ausgeteilten Prüfungsunterlagen (drei Blätter, bestehend aus 1 Deckseite, 4 Aufgabenseiten mit insgesamt 6 Aufgaben und eine leere Seite)!
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. (21 Punkte) Geben Sie zu den folgenden Ausdrücken jeweils den Typ (1 Punkt) und den Wert (2 Punkte) an. Es gibt keine Punkte für die Teilschritte der Auswertung. Sie brauchen diese nicht hinzuschreiben.

Zu Beginn jeder der Teilaufgaben gilt: Die Variable `x` ist vom Typ `int` und hat den Wert 8. Die Variable `b` ist vom Typ `bool` und hat den Wert `true`.

a) `6 / 3 / 4 + 1.5`

b) `x != 0 && 7 / x == 0 && x / 7.0 > 1`

c) `7.2f - 4 > 4 && !b || b`

d) `b == false && ++x >= 9`

e) `2.0f + x++ - 2.5 * 3`

f) `x % 2 % 2 - x`

g) `x-- % 4 == 2 && 128 % 2 == 0`

Aufgabe 2. (12 Punkte) Geben Sie für die folgenden drei Schleifenkonstrukte jeweils die Folge der ausgegebenen Zahlen an!

a)
```
for (int i=1023; i>0; i/=2)
     std::cout << (i%2) << " ";
```

b)
```
int n=13;
while (n>1) {
  std::cout << n << " ";
  if (n%2==0)
    n/=2;
  else
    n=3*n+1;
}
```

c)
```
int i=1;
int j=1;
while (i<=100) {
  std::cout << i << " ";
  j+=2;
  i+=j;
}
```

Aufgabe 3. (20 Punkte) Die Binärdarstellung einer natürlichen Zahl $n \geq 0$ ist die eindeutige Bitfolge $(b_0, b_1, \ldots)$ mit $b_i \in \{0, 1\}$ für alle i, so dass

$$\sum_{i=0}^{\infty} b_i 2^i = n.$$

Nur endlich viele Bits haben dabei den Wert 1. In der Binärdarstellung von n ist das Bit i gesetzt, falls $b_i = 1$ gilt.

Geben Sie eine Definition der folgenden Funktion an.

```
// POST: gibt genau dann true zurueck, wenn in der Binaerdarstellung
//       von n das Bit i gesetzt ist.
bool has_bit (unsigned int n, unsigned int i);
```

Aufgabe 4. (8 / 15 Punkte)

a) Finden und beheben Sie vier Fehler in folgendem Programm, das eine Matrix M der Grösse `dim`×`dim` erzeugt und sie mit der Einheitsmatrix initialisiert. (Sei m_{ij} der Eintrag von M in Zeile i und Kolonne j; für die Einheitsmatrix gilt $m_{ij} = 1$ falls $i = j$, und $m_{ij} = 0$ andernfalls.)

```
int main()
{
  unsigned int dim = 10;
  double M[dim][dim];

  // initialisiere M mit der Einheitsmatrix
  for (int i=1; i<=dim; ++i)
    for (int j=1; j<=dim; ++j)
      if (i=j)
        M[i][j]=1.0;
      else
        M[i][j]=0.0;

  // mache irgendetwas mit M

  return 0;
}
```

b) Erweitern Sie das korrigierte Programm um einen Abschnitt, der die Matrix M transponiert, d.h. Eintrag m_{ij} mit Eintrag m_{ji} vertauscht, für alle i, j:

```
  // mache irgendetwas mit M

  // transponiere M
  // hier Ihre Erweiterung
```

Aufgabe 5. (25 Punkte) Betrachten Sie folgende rekursive Funktion, die das Problem der Türme von Hanoi löst.

```
// PRE:  von, nach, ueber sind verschieden
// POST: gibt Zuege aus, mit denen die obersten n
//       Scheiben vom Stab "von" zum Stab "nach" gebracht
//       werden koennen, unter Benutzung des Stabs "ueber"
void hanoi (unsigned int n, int von, int nach, int ueber) {
  if (n == 0)
    // nichts zu tun
    return;
  else {
    // 1. Die obersten n-1 Scheiben vom Start- zum Hilfsstab bringen
    hanoi (n-1, von, ueber, nach);
    // 2. Die oberste Scheibe vom Start- zum Zielstab bringen
    std::cout << "(" << von << "," << nach << ")";
    // 3. Die obersten n-1 Scheiben vom Hilfs- zum Zielstab bringen
    hanoi (n-1, ueber, nach, von);
  }
}
```

Wie gross ist die Anzahl der Züge (= Anzahl der `std::cout`-Anweisungen) bei einem Aufruf von `hanoi(n, von, nach, ueber)`, in Abhängigkeit von n?

Hinweis: Definieren Sie $T(n)$ als die gesuchte Anzahl der Züge bei n Scheiben und leiten Sie eine Rekursionsgleichgung für $T(n)$ her! "Raten" Sie $T(n)$, indem Sie kleine Fälle oder die Rekursionsgleichung anschauen und verwenden Sie dann vollständige Induktion, um den geratenen Wert für $T(n)$ zu beweisen.

Aufgabe 6. (10 / 10 Punkte) Betrachten Sie das untenstehende Programm und beantworten Sie die folgenden beiden Fragen dazu.

a) Beschreiben Sie in Worten die Funktionalität der Klasse `A`!

b) Was ist die Ausgabe des Programms? Begründen Sie kurz, wie sich diese Ausgabe aufgrund der Funktionalität aus Teil a) ergibt!

```
#include<cassert>
#include<iostream>

class A {
private:
  double       s;
  unsigned int n;

public:
  A()
    : s(0), n(0)
  {}

  void i (const double d)
  {
    s+=d;
    ++n;
  }

  double a() const
  {
    assert(n>0);
    return s/n;
  }
};

int main()
{
  A x;
  x.i(3);
  x.i(4);
  std::cout << x.a();
  return 0;
}
```