Prüfung — Informatik D-MATH/D-PHYS
9. 8. 2010 **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 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. (20 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:

```
double x[2] = {2.0,1.0};
double* p = &x[0];
int a = 1;
int b = 0;
bool z = false;
```

Beachten Sie, dass der Typ unter Umständen auch "Zeiger auf ..." sein kann, wobei die Punkte für den zugrundeliegenden Typ (`int`, `double` oder `bool`) stehen. In diesem Fall geben Sie anstelle des Wertes an, auf welchen anderen L-Wert der Zeiger zeigt. Zu Beginn gilt zum Beispiel: `p` ist vom Typ "Zeiger auf `double`" und zeigt auf `x[0]`.

a) `x[0] * 2.5 / 2`

b) `111 % 10 - a * 2`

c) `z == false`

d) `x[0] = b`

e) `z && b == 1 || a == 1`

f) `&*p`

g) `*p = a + 3`

h) `&x[0] == p`

i) `a == *(p = &x[1])`

j) `++p`

Aufgabe 2. (10 / 10 Punkte) Das Ziel dieser Aufgabe ist es, zu beweisen, dass es reelle Zahlen gibt, die von keinem C++ Programm berechnet werden können. Eine Zahl kann von einem Programm berechnet werden, wenn das Programm die Zahl bis zu einer beliebigen (endlichen) Genauigkeit ausgeben kann. Wir führen den Beweis in zwei Schritten.

a) Beweisen Sie, dass die Menge der reellen Zahlen, $\mathbb{R}$, überabzählbar ist. Das heisst konkret, zeigen Sie, dass es keine Bijektion zwischen der Menge $\mathbb{R}$ und der Menge der natürlichen Zahlen, $\mathbb{N}$, gibt.

   Tipp: Sie haben den entsprechenden Beweis in der Vorlesung gesehen.

b) Zeigen sie, dass es zwischen der Menge aller C++ Programme und $\mathbb{N}$ eine Bijektion gibt.

Teilaufgaben a) und b) zusammen implizieren das Gewünschte, denn es kann keine Bijektion zwischen der Menge aller Programme und den reellen Zahlen geben.

Aufgabe 3. (10 Punkte) Implementieren Sie die Funktion `print_stars`, die auf dem Ausgabestrom `std::cout` 2^n Sterne ausgibt.

```
// POST: Gibt auf dem Standardausgabestrom (std::cout) 2^n Sterne aus.
void print_stars(unsigned int n);
```

Aufgabe 4. (8 / 3 / 8 / 3 / 3 Punkte) Folgender Quelltext implementiert ein Sortierverfahren. Die Variable `n` hat den Typ `int` und speichert eine positive ganze Zahl. Es geht darum die `double`-Werte in dem Feld `c` aufsteigend zu sortieren. Die Werte `c[0],...,c[n-1]` sind korrekt initialisiert.

```
1:  for (int i = 1; i < n; ++i) {
2:     int index = i;
3:     int value = c[index];
4:     while (index > 0 && value < c[index-1]) {
5:        c[index] = c[index-1];
6:        --index;
7:      }
8:      c[index] = value;
9:   }
```

a) Beschreiben Sie, wie das Sortierverfahren funktioniert. Sie können Ihre Erklärungen auch illustrieren.

b) Wie oft wird die Anweisung in Zeile `8` ausgeführt? Geben Sie Ihre Antwort in Abhängigkeit von `n` an.

c) Wie oft wird der Vergleich `value < c[index-1]` in Zeile `4` *maximal* ausgeführt? Geben Sie Ihre Antwort wiederum in Abhängigkeit von `n` an.

d) Geben Sie eine mögliche Eingabebelegung des Feldes `c` an, für die die maximale Anzahl an Vergleichen der Form `value < c[index-1]` erreicht wird.

e) Gibt es noch effizientere Sortierverfahren? Falls ja, nennen Sie ein Beispiel und geben Sie in Abhängigkeit von `n` an, wie viele Vergleiche dabei maximal notwendig sind. Sie müssen Ihre Aussagen nicht beweisen.

Aufgabe 5. (6 / 6 / 4 / 9 Punkte) Gegeben sei folgende Implementierung einer Klasse mit dem Namen `Clock`.

```
01: class Clock {
02:   public:
03:     Clock(const unsigned int h,
04:           const unsigned int m,
05:           const unsigned int s) const
06:           : h_(h), m_(m), s_(s)
07:     {
08:     }
09:
10:     tick()
11:     {
12:       h_ += (m_ += (s_ += 1) / 60) / 60;
13:       h_ %= 24;
14:       m_ %= 60;
15:       s_ %= 60;
16:     }
17:
18:     void time(unsigned int& h,
19:               unsigned int& m,
20:               unsigned int& s) const
21:     {
22:       h = h_;
23:       m = m_;
24:       s = s_;
25:       tick();
26:     }
27:
28:   private:
29:     unsigned int h_;
30:     unsigned int m_;
31:     unsigned int s_;
32: };
```

a) Beschreiben Sie in kurzen Sätzen den Zweck der Mitgliedervariablen `h_`, `m_` und `s_`, der Methoden `tick()` und `time(...)` und des Konstruktors `Clock(...)`.

b) In die Implementierung der Klasse haben sich drei Fehler eingeschlichen, die schon beim Kompilieren eine Fehlermeldung erzeugen. Benennen Sie an welcher Stelle die Fehler sind und erklären Sie kurz, was das Problem ist.

c) Der Funktionsrumpf des Konstruktors ist leer. Heisst das, dass der Konstruktor gar nichts tut? Erläutern Sie Ihre Antwort in maximal zwei Sätzen.

d) Implementieren Sie die folgende globale Funktion `operator<<`, die auf einen beliebigen Ausgabestrom `o` mit Hilfe einer Referenz auf eine Instanz `c` der Klasse `Clock` die Zeit ausgibt.

```
// POST: Die Zeit, die in c gespeichert ist,
//       wurde im Format h:m:s auf o ausgegeben.
std::ostream& operator<< (std::ostream& o, const Clock& c);
```

Bemerkung: Sie müssen die Fälle, in denen eine der Variablen eine einstellige Dezimalrepräsentation hat, nicht gesondert behandeln. Konkret heisst das, dass die Ausgabe `1:2:20` ausreichend ist, obwohl das gebräuchliche Format eigentlich `01:02:20` wäre.

Aufgabe 6. (14 / 6 Punkte) Gegeben sei die folgende Potenzreihe, die es uns ermöglicht, den Sinus einer Zahl x zu approximieren.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots,$$

wobei $x \in \mathbb{R}$.

a) Implementieren Sie die folgende Funktion, die den Sinus einer Zahl berechnet. Berechnen Sie dazu die Potenzreihe bis und mit dem zwanzigsten Term.

```
// POST: Es wird eine Approximation von sin(x) berechnet, die die
//       ersten 20 Terme der Potenzreihe beruecksichtigt.
double sinus(double x);
```

b) Was ist, wenn der Eingabewert sehr gross (z.B. $x = 10'000$) ist? Liefert Ihre Funktion dann auch gute Resultate? Begründen Sie! Falls nein, sehen Sie eine Möglichkeit, das Problem einfach zu beheben?