Prüfung — Informatik D-MATH/D-PHYS
6. 08. 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 (zwei beidseitig bedruckte Blätter, bestehend aus 1 Deckseite und 3 Aufgabenseiten 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: **Wörterbücher sind erlaubt; sonst keine**.
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 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. (20 Punkte) Betrachten Sie die folgenden vollständig geklammerten `C++` Ausdrücke. Geben Sie an, welche Klammerpaare redundant sind, das heisst welche Klammerpaare weggelassen werden können, so dass der Ausdruck trotzdem noch gleich ausgewertet wird. Für die vorliegenden Ausdrücke (nicht für Ihre gekürzte Version) geben Sie zudem die Auswertung inklusive aller Zwischenschritte an. Bezeichnen Sie sowohl den Wert als auch den Typ der jeweiligen Zwischenresultate.

(a) `(99/(1.5/(3/2)))`

(b) `((1-((-5)>(+3)))==1)`

Hinweis: Bei der Auswertung beginnen Sie am besten für jeden separaten Schritt jeweils eine neue Zeile, unterstreichen den soeben ausgewerteten Teil und schreiben dessen Typ darunter.

Aufgabe 2. (15 Punkte) Betrachten Sie die folgenden zwei Funktionen.

```
// PRE: n > 0
unsigned int f (unsigned int n) {
   if (n < 10) return 1;
   return 1 + f(n / 10);
}

// PRE: n > 0
void g (unsigned int n, unsigned int k) {
    int c = f(n);
    if (k < c) return;
    for (int i = 0; i < k - c; ++i) {
        std::cout << " ";
    }
    std::cout << n;
}
```

Geben Sie die Nachbedingungen der Funktionen `f` und `g` an. Die Nachbedingungen müssen die Rückgabewerte und Effekte der Funktionen in Abhängigkeit von ihren Parametern vollständig charakterisieren.
Hinweis: Sie können davon ausgehen, dass `iostream` korrekt eingebunden wurde.

Aufgabe 3. (20 Punkte) Nehmen Sie an, dass das Feld

```
int a[100];
```

vollständig mit ganzen Zahlen intialisiert wurde. Im Sichtbarkeitsbereich von `a` wird nun folgende Implementation von Bubble Sort ausgeführt.

```
01: bool switch = true;
02: for(int i = 0; switch; ++i) {
03:    switch = false;
04:    for(int j = 0; j < 100 - i; ++j) {
05:       if (a[j] > a[j+1]) {
06:          int tmp = a[j];
07:          a[j] = a[j+1];
08:          a[j+1] = tmp;
09:          switch = true;
10:       }
11:    }
12: }
```

(a) Das obige Fragment führt zu einem Absturz des Programmes mit der Fehlermeldung "`Segmentation fault`", was so viel bedeutet wie Zugriffsfehler. Beschreiben Sie, wo das Problem liegt, und schlagen Sie eine Korrektur vor, so dass das Programmfragment die Zahlen in `a` fehlerfrei aufsteigend sortiert.

Hinweis: Es gibt mehrere Möglichkeiten, den Fehler zu beheben.

(b) Wie oft wird der Vergleich in Zeile `05` ausgeführt? Betrachten Sie die korrigierte Variante, die korrekt sortiert! Begründen Sie ihre Antwort.

Aufgabe 4. (15 Punkte) Implementieren Sie die folgende Funktion, welche genau dann `true` zurückgibt, wenn die Primfaktorenzerlegung von `x` keine Zahlen enthält, die drei oder mehr Stellen haben.

```
// POST: returns true if and only if no prime divisor of x
//       is larger than 99.
bool petty_number (unsigned int x);
```

Aufgabe 5. (20 Punkte) Zeigen Sie auf, dass die Menge aller möglichen (syntaktisch korrekten) `C++` Programme gleich mächtig ist wie die Menge $\mathbb{N}$. Erläutern Sie auch, wie es (zumindest theoretisch) möglich ist, alle syntaktisch korrekten Programme zu generieren.

Aufgabe 6. (30 Punkte) Ihre Aufgabe ist es, eine C++ Klasse zu implementieren, mit der man 10×10 Matrizen über den ganzen Zahlen speichern und bearbeiten kann.

(a) Schreiben Sie eine Klasse `Matrix10`, die eine geeignete Datenstruktur enthält, um alle Einträge der Matrix zu speichern.

(b) Schreiben Sie einen Konstruktor für die Klasse, die die Matrix mit Nullen initialisiert. Das heisst alle Einträge der Matrix sollen Null sein.

(c) Implementieren Sie die folgenden Mitgliedsfunktionen, die dem Elementzugriff dienen. Beachten Sie, dass die Zeilen- resp. Spaltennummerierung der Matrix mit 0 beginnt.

```
// PRE: i < 10, j < 10
// POST: Returns the value of the matrix at row i, column j.
int getEntry(unsigned int i, unsigned int j);

// PRE: i < 10, j < 10
// POST: The value of the matrix at row i, column j, is set
//       to n.
void setEntry(unsigned int i, unsigned int j, int n);
```

(d) Implementieren Sie eine Mitgliedsfunktion, die testet, ob die gegebene Matrix symmetrisch ist.

```
// POST: Returns true if and only if the matrix is symmetric, i.e.
//       if the entry at row i, column j, is equal to the entry at
//       row j, column i, for all choices of i and j.
bool symmetric();
```