Prüfung Informatik D-MATH/D-PHYS
29. 9. 2006 9:00-11:00

Dr. Bernd Gärtner

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 (2 beidseitig bedruckte Blätter und 1 einseitig bedrucktes Blatt, bestehend aus 1 Deckseite und 4 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: **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!
6. Sie dürfen die 6 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 3 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) Geben Sie für jeden der folgenden Ausdrücke Typ und Wert an! Bei Fliesskommaarithmetik nehmen wir den IEEE Standard 754 an. Sie können die Antwort direkt auf das Aufgabenblatt schreiben (erste Kolonne: Typ, zweite Kolonne: Wert), oder Sie verwenden ein separates Blatt.

Aufgabe	Ausdruck	Typ	Wert
(a)	`2e2-3e3f>-23.0`		
(b)	`-7+7.5`		
(c)	`1.0f/2+1/3+1/4`		
(d)	`true||false&&false`		
(e)	`1u-2u<0`		
(f)	`1+2*3+4`		
(g)	`int(8.5)-int(7.6)`		
(h)	`100*1.1==110`		
(i)	`10*11.0==110`		
(j)	`4+12/4.0/2`		

Aufgabe 2. (25 Punkte) Eine natürliche Zahl $n \in \{0, 1, 2 \ldots\}$ heisst *Quadratsumme*, falls sie sich in der Form $n = a^2 + b^2$ schreiben lässt, wobei a und b ebenfalls natürliche Zahlen sind. Zum Beispiel sind $13 = 3^2 + 2^2$ und $16 = 4^2 + 0^2$ Quadratsummen, während 14 keine Quadratsumme ist, wie man durch Ausprobieren aller Möglichkeiten leicht sieht. Implementieren Sie eine Funktion

```
// POST: gibt genau dann true zurueck, wenn n eine Quadratsumme ist.
//       Falls der Rueckgabewert true ist, so gilt n = a^2 + b^2
bool is_square_sum (unsigned int n, unsigned int& a, unsigned int& b);
```

die für eine gegebene natürliche Zahl n testet, ob diese eine Quadratsumme ist. Falls n eine Quadratsumme ist, soll die Funktion die Werte der Parameter `a` und `b` so setzen, dass $n = a^2 + b^2$ gilt.

Aufgabe 3. (20 Punkte) Betrachten Sie die folgenden zwei Funktionen

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

```
unsigned int g (unsigned int n) {
  if (n < 10) return n;
  unsigned int r = n % 10;
  unsigned int k = g (n / 10);
  return r * f(n) + k;
}
```

Geben Sie die Nachbedingungen der Funktionen `f` und `g` an! Die Nachbedingungen müssen die jeweiligen Rückgabewerte der Funktionen in Abhängigkeit von ihren Parametern vollständig charakterisieren.

Aufgabe 4. (10 Punkte) Was ist die Ausgabe des folgenden Programms? Leiten Sie diese Schritt für Schritt her, indem Sie jede Zeile, die einer Variablen einen Wert zuweist, mit dem entsprechenden Wert annotieren. Für die Zeilen 3–5 sind diese Werte bereits eingetragen.

```
0:  #include<iostream>
1:
2:  int main() {
3:    int i = 1;                                  // 1
4:    int k = 2;                                  // 2
5:    int l = 0;                                  // 0
6:    {
7:      int k = i;
8:      l = k;
9:      {
10:       int i = 3;
11:       int k = i;
12:       l += k;
13:     }
14:     int l = 5;
15:     l += k;
16:     i += l;
17:   }
18:   l += i;
19:
20:   std::cout << l << std::endl;
21: }
```

Aufgabe 5. (30 Punkte) Sei $\{0, 1, \ldots, U-1, U\}$ der Wertebereich des Typs `unsigned int`. Wir wollen eine Variante des Typs `int` implementieren, die den Wertebereich

$$\{-U, -U+1, \ldots, -1, 0, 1, \ldots, U-1, U\}$$

hat. Dazu repräsentieren wir jeden Wert in diesem Bereich durch einen Wert vom Typ `unsigned int` und ein separates Vorzeichen. Der resultierende Typ `Int` habe unter anderem folgende öffentliche Member-Funktionen:

```
class Int {
public:
  // POST: *this wurde mit x initialisiert.
  Int (int x);

  // POST: *this wurde initialisiert mit
  //       x, falls  negative == false
  //      -x, falls  negative == true
  Int (unsigned int x, bool negative);

  // POST: Rueckgabewert ist -*this
  Int operator-() const;

  // POST: y wurde zu *this addiert.
  Int operator+=(const Int& y);

  // POST: y wurde von *this subtrahiert.
  Int operator-=(const Int& y);

private:
  ...
};
```

(a) Ergänzen Sie den `private`-Teil, indem Sie eine geeignete Repräsentation (im Hinblick auf Teil (b)) wählen. (4 Punkte)

(b) Implementieren Sie (unter Verwendung der Repräsentation aus (a)) die fünf Member-Funtionen

- `Int::Int (int x)` (4 Punkte)
- `Int::Int (unsigned int x, bool negative)` (4 Punkte)
- `Int Int::operator-() const` (4 Punkte)
- `Int& Int::operator+=(const Int& y)` (7 Punkte)
- `Int& Int::operator-=(const Int& y)` (7 Punkte)

Über- und Unterlauf sollen dabei nicht behandelt werden.

Hinweis: Member-Funktionen können natürlich andere Member-Funktionen aufrufen, und auf diese Weise können Sie sich möglicherweise Arbeit sparen.

Aufgabe 6. (15 Punkte) Finden Sie die Fehler im folgenden Programm und geben Sie eine korrigierte Version des Programms an! Beschreiben Sie die Ausgabe des korrigierten Programms in Abhängigkeit von `n` (Skizze genügt)!

```
0:  #include<cmath>
1:
2:  int main() {
3:    std::cout << n =? ;
4:    int n;
5:    std::cin >> n;
6:
7:    for (int i=0, i<n, ++i) {
8:      for (int j=0, j<n, ++j)
9:        if ( i-j = 0 || i+j = n-1)
10:         std::cout << "*";
11:       else
12:         std::cout << " ";
13:     std::cout << "\n";
14:   }
15:   return 0
16: }
```