Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

19. Dezember 2006

Vorlesung "Objektorientierte Programmierung" — Probeklausur I: Musterlösung —

Die Klausur ist schlecht ausgefallen. Bis zur richtigen Klausur gibt es noch viel zu tun:

Gesamtpunkte	Anzahl	Note
10	0	1.0
9	1	1.0 oder 1.3
8	2	ca. 2.0
7	2	2.3 oder 2.7 oder 3.0
6	15	3.0 bis 4.0
5	7	4.0 oder 5.0
4	11	5.0
3	6	5.0

Die Noten dienen nur als Anhaltspunkt. Es ist nicht so zu verstehen, daß es bei der richtigen Klausur garantiert die gleichen Noten gibt. Es ist aber garantiert, daß Sie mit 60% der Punkte bestehen, und mit 95% der Punkte eine 1.0 bekommen (diese Grenzen werden möglicherweise leicht gesenkt).

Aufgabe 1 (Arithmetischer Ausdruck) 1 Punkt

Was gibt das folgende Programm aus?

```
#include <iostream>
using namespace std;

int main()
{
        int i = 7;
        int j = 10;

        int n = 5 + 3 * i % j << 1;
        cout << n;

        return 0;
}
```

Lösung:

Das Programm gibt 12 aus. Dazu muß man die Zuweisung

```
int n = 5 + 3 * i % j << 1;
```

analysieren (der Rest ist ja sehr einfach).

Die Operatoren höchster Priorität sind `*` und `%`. Sie haben die gleiche Priorität, also wird von links geklammert (Ausnahmen sind Zuweisungen und Präfixoperatoren). Dies ergibt:

```
int n = 5 + ((3 * i) % j) << 1;
```

Anschließend steht das `+` in der Prioritätenliste:

```
int n = (5 + ((3 * i) % j)) << 1;
```

Nun kommt der Shift-Operator `<<`:

```
int n = ((5 + ((3 * i) % j)) << 1);
```

Die Zuweisung wird dann ganz am Ende ausgeführt. Da `i` den Wert `7` hat, ist `3 * i` $= 21$. Anschließend wird der Divisionsrest bei Division durch `j` $= 10$ bestimmt: ist $21\%10 = 1$. Dann wird 5 addiert: $5 + 1 = 6$. Zum Schluß findet ein Linksshift um eine Bitposition statt. Das entspricht einer Multiplikation mit 2, daher ist das Ergebnis 12. Dieser Wert wird `n` zugewiesen und `n` anschließend ausgedruckt.

Wenn man will, kann man sich den Linksshift auch auf Bitebene anschauen. Der Wert 6 ist intern repräsentiert als 0110 (die beiden 1-Bits haben den Wert 4 und den Wert 2). Nach dem Linksshift hat man 1100. Hier haben die beiden 1-Bits den Wert 8 und den Wert 4. Daher erhält man tatsächlich 12.

Statistik:

16 von 44 Teilnehmern hatten diese Aufgabe korrekt (36%).

Aufgabe 2 (Logische Bedingung) **1 Punkt**

Was gibt das folgende Programm aus?

```
#include <iostream>
using namespace std;

int main()
{
        int i = 7;
        int j = 10;

        if(i == j)
                cout << "A";
        else if(i > j || j < 20)
                cout << "B";
        else if(i && j >= i)
                cout << "C";
        else
                cout << "D";

        return 0;
}
```

Lösung:

Man muß bei der `if-else if`-Kette die verschiedenen Bedingungen nacheinander prüfen, die Anweisung unter der ersten korrekten Bedingung wird ausgeführt.

- `i == j` ist falsch, denn `i` hat den Wert `7` und `j` hat den Wert `10`.
- `i > j || j < 20` ist eine Disjunktion (logisches "oder"). Die Teilbedingung `i > j` ist falsch, aber `j < 20` ist wahr. Bei der Disjunktion reicht es, wenn eine der Teilbedingungen wahr ist. Also ist die Gesamtbedingung wahr, und es wird `B` ausgedruckt.
- Weitere Bedingungen werden gar nicht mehr ausgewertet, da sie unter dem `else` stehen. Falls es Sie aber interessiert: `&&` ist die Konjunktion (logisches und). Eine solche Bedingung ist nur erfüllt, wenn beide Teilbedingungen zutreffen. `i` zählt als logisch wahr, da es einen von Null verschiedenen Wert hat. Die Teilbedingung `j >= i` ist auch wahr. Also wäre die Gesamtbedingung `i && j >= i` wahr. Aber, wie gesagt, die CPU springt direkt von `cout << "B"` zu der `return`-Anweisung.

Statistik:

38 von 44 Teilnehmern hatten diese Aufgabe korrekt (86%).

Aufgabe 3 (Variablen, Pointer, Referenzen) 1 Punkt

Was gibt dieses Programm aus?

```
#include <iostream>
using namespace std;

int main()
{
        int i = 1;
        int j = 2;
        int a[3] = { 3, 4, 5 };
        int &r = i;
        int *p = &j;
        int *q = a;
        r++;
        a[i] = j;
        *(q+2) += *p;
        cout << a[0] << "," << a[1] << "," << a[2] << "\n";
        return 0;
}
```

Lösung:

Man muß dieses Programm Zeile für Zeile durchgehen und über die Werte der Variablen Buch führen:

- Nach den Deklarationen mit Initialisierung sieht die Situation so aus:

i	j	a[0]	a[1]	a[2]	r	p	q
1	2	3	4	5	=i (1)	→j	→a[0]

- Dann kommt `r++`. Es ist wichtig, zu verstehen, daß die Referenz `r` nur ein anderer Name für `i` ist. Diese Anweisung erhöht also `i` (und damit indirekt natürlich auch `r`). Man kann sich die Referenz wie einen Pointer vorstellen, der immer automatisch dereferenziert wird.

i	j	a[0]	a[1]	a[2]	r	p	q
2	2	3	4	5	=i (2)	→j	→a[0]

- `a[i] = j` setzt damit `a[2]` auf 2:

i	j	a[0]	a[1]	a[2]	r	p	q
2	2	3	4	2	=i (2)	→j	→a[0]

- Nun kommt `*(q+2) += *p`. Da `q` auf `a[0]` zeigt, zeigt `q+2` auf `a[2]`. Der Zeiger wird mit `*` derefenziert, damit hat man also die Variable `a[2]` selbst. Der Zeiger `p` zeigt auf `j`, `*p` ist also die Variable `j`, von der in diesem Kontext (auf der rechten Seite einer Zuweisung) der Wert genommen wird, das ist `2`. Dieser Wert wird durch `+=` auf `a[2]` aufaddiert.

i	j	a[0]	a[1]	a[2]	r	p	q
2	2	3	4	4	=i (2)	→j	→a[0]

Daher wird `3, 4, 4` ausgegeben.

Statistik:

6 von 44 Teilnehmern hatten diese Aufgabe korrekt (14%).

Aufgabe 4 (Programm) 5 Punkte

Schreiben Sie ein Programm, das eine positive Zahl einliest, und dann eine "Pyramide" (oder "Tannenbaum") der entsprechenden Höhe ausgibt. Für die Eingabe 3 soll z.B. folgende Ausgabe erscheinen (drei Zeilen):

```
    *
  * * *
* * * * *
```

Für die Eingabe 1 würde entsprechend nur ein einzelner Stern gedruckt. Da in der Vorlesung die Ein-/Ausgabe noch nicht gründlich besprochen wurde, dürfen Sie voraussetzen, daß der Benutzer korrekt eine positive Zahl eingibt (und auch kein sonstiger Fehler bei der Eingabe auftritt). Bemühen Sie sich um Verständlichkeit und guten Programmierstil. Schreiben Sie das Programm vollständig und möglichst ohne Syntaxfehler.

Lösung:

```
#include <iostream>
using namespace std;

int main()
{
        int hoehe;
        cout << "Bitte die Hoehe des Baums eingeben: ";
        cint >> hoehe;
        cout << '\n';

        for(int i = 1; i <= hoehe; i++) {
                int leerzeichen = hoehe - i;
                int sterne = 2 * i - 1;
                for(int j = 1; j <= leerzeichen; j++)
                        cout << ' ';
                for(int k = 1; k <= sterne; k++)
                        cout << '*';
                cout << '\n';
        }

        return 0;
}
```

- Natürlich gibt es auch viele andere Möglichkeiten. Viele der Abgaben waren nicht gut zu verstehen, bei der Endklausur können dann Punkte abgezogen werden. Bemühen Sie sich um eine möglichst einfache und übersichtliche Lösung.
- Mindestens für die Variable, die die Höhe des Baums enthält, sollte man einen selbstdokumentierenden Varibablennamen wählen. `i` oder `n` ist dafür ein schlechter Name.

- Solange man nur einfach zählen will, sollte man eine `for`-Schleife verwenden, und keine `while`-Schleife. Die `for`-Schleife ist übersichtlicher, weil da die ganze Schleifenkontrolle an einer Stelle zusammen steht. Man kann also sofort übersehen, wie häufig und für welche Werte die Schleife ausgeführt wird.
- Wen man nur einzelne Zeichen ausgeben will, wie z.B. den Stern, ist es etwas effizienter, `'*'` zu schreiben statt `"*"`. Im zweiten Fall wird die Ausgabefunktion für Strings verwendet, die eine Schleife enthält, in der jedes Zeichen einzeln ausgegeben wird. Wenn man dagegen `'*'` schreibt, wird direkt die Ausgabefunktion für einzelne Zeichen aufgerufen. Die Schleifenkontrolle entfällt so.
- Leider war bei der Aufgabe nicht klar, ob zwischen den Sternchen Leerzeichen ausgegeben werden sollen. Das war eigentlich nicht beabsichtigt, aber die Beispielausgabe sieht doch so aus. Falls Sie das getan haben, wäre das natürlich auch korrekt. In diesem Fall müssen Sie aber auch zu Anfang die doppelte Anzahl Leerzeichen ausgeben. Bedauerlicherweise ist mir erst bei der vorletzten Klausur aufgefallen, daß es auch diese Interpretationsmöglichkeit gibt. Wenn es eine richtige Klausur gewesen wäre, hätte ich alle Klausuren nochmal anschauen müssen, ob ich deswegen etwa einen Punkt zuviel abgezogen habe. Da es nur eine Probeklausur war, habe ich das nicht getan.
- Wenn Sie irgendetwas von den Korrekturen nicht verstehen, oder ich einen Fehler gemacht habe (z.B. mit den Leerzeichen zwischen den Sternchen), melden Sie sich bitte.

Statistik:

Die Punkteverteilung war:

Punkte	Anzahl
6	0
5	1
4	6
3	6
2	10
1	19
0	2

Wie oben erklärt, ist es möglich, daß ich in einzelnen Fällen einen Punkt zu viel abgezogen habe. Auch so hätte aber nur 1/44 das Programm vollständig korrekt. Dabei habe ich nur in ganz extremen Fällen Punkte wegen schlechtem Stil abgezogen. Ich habe auch nichts abgezogen, wenn die Eingabeaufforderung fehlte etc. Die echte Klausur wird strenger korrigiert.

Aufgabe 5 (Fehlersuche) 2 Punkte

Das folgende Programm enthält (mindestens) 3 Fehler, die der Compiler finden sollte. Bitte geben Sie zwei dieser Fehler an (es ist möglicherweise schwierig, alle drei zu entdecken). Sie bekommen keinen Extrapunkt, wenn Sie alle drei finden. Falls Sie mehr als zwei Fehler angeben, werden nur die ersten beiden gewertet. Geben Sie bitte die Zeilennummer mit an, in der der Compiler den Fehler finden müßte, wenn er das Programm von oben nach unten liest. Die in Klammern vorangestellten Zeilennummern sind natürlich nicht Bestandteil des Programms (und zählen nicht als Fehler).

```
 (1)  #include <iostream>
 (2)  using namespace std;
 (3)
 (4)  int main()
 (5)  {
 (6)          cin >> i;
 (7)          if(i != 0)
 (8)                  cout << "i ist ";
 (9)                  cout << i << ".\n";
(10)          else
(11)                  cout << "i ist null.\n";
(12)  }
```

Lösung: Die drei Fehler waren:

- In Zeile 6 ist die Variable i nicht deklariert. In Zeile 7 und 9 dann natürlich auch nicht, aber ich habe diesen Fehler nur einmal gezählt. Manche Studenten haben geschrieben, die Variable wäre nicht initialisiert. Das ist das falsche Wort: Initialisierung ist die erstmalige Wertzuweisung, das geschieht korrekt durch die Leseanweisung in Zeile 6. Die Deklaration teilt dem Compiler dagegen mit, daß Sie eine Variable `i` vom Typ `int` verwenden wollen.
- In Zeile 10 steht ein `else`, zu dem es kein passendes `if` mehr gibt. Zwar steht in Zeile 7 ein `if`, aber da um die abhängigen Anweisungen keine `{...}` stehen, hängt formal nur die erste eingerückte Anweisung von dem `if` ab. Die nächste Anweisung folgt dann dem `if` und wird immer ausgeführt, unabhängig von der Bedingung. Damit ist das `if` dann aber sozusagen “geschlossen”, und man kann kein `else` mehr angeben.

  Der Compiler findet den Fehler erst in Zeile 10, bis dahin ist alles legal. Allerdings stimmt es natürlich, daß die Einrückung und sonstige Semantik des Programms sehr nahelegt, daß der eigentliche Fehler die fehlenden geschweiften Klammern sind. Ich war in der Anerkennung von Zeilennummern für diesen Fehler daher sehr großzügig.
- Am Ende von `main` fehlt eine `return`-Anweisung. Die Funktion soll ja einen `int`-Wert zurückliefern, aber es ist keiner definiert.

Inzwischen habe ich allerdings gelernt (aus dem Buch von Stroustrup, der muß es ja wissen), daß in diesem Fall `main` den Wert `0` zurückliefert (diese spezielle Regelung gilt nur für `main`, nicht für andere Funktionen). Dann wäre es also kein Fehler (ich habe ihn aber selbstverständlich anerkannt, weil es in der Vorlesung ja so unterrichtet wurde).

Zu meiner Verteidigung möchte ich noch vorbringen, daß mein Microsoft Visual C++ 6.0 Compiler eine Warnung ausgibt, wenn man das `return` wegläßt. Die Programmierer bei Microsoft waren also wohl auch der Meinung, daß es ein Fehler ist. Dann wäre möglich, daß mindestens bei diesem Compiler der Rückgabewert in diesem Fall undefiniert ist.

Statistik:

Die Punkteverteilung war:

Punkte	Anzahl
2	35
1	9
0	0

Umfrage

34 Teilnehmer (72%) haben gesagt, daß sie vor dieser Vorlesung mindestens etwas programmieren konnten. An Sprachen wurden genannt:

Pascal	22
Delphi	10
C++	10
PHP	5
Basic	4
Java	4
JavaScript	2
Oberon	2
Prolog	1
Haskel	1

Pobeklausurergebnisse in Abhängigkeit der Antwort auf die Frage "Wieviel Prozent des Vorlesungsstoffs ist für Sie neu?":

- 2 Teilnehmer (5%) haben 0% angekreuzt.

Gesamtpunkte	Anzahl
9	1
7	1

- 9 Teilnehmer (20%) haben 25% angekreuzt.

Gesamtpunkte	Anzahl
6	7
4	2

- 8 Teilnehmer (18%) haben 50% angekreuzt.

Gesamtpunkte	Anzahl
8	1
6	4
5	1
4	2

- 15 Teilnehmer (34%) haben 75% angekreuzt.

Gesamtpunkte	Anzahl
8	1
7	1
6	4
5	3
4	4
3	2

- 10 Teilnehmer (23%) haben 100% angekreuzt.

Gesamtpunkte	Anzahl
5	3
4	3
3	4

Da an den Übungen 84 Studierende teilnehmen, und in der Vorlesung nur 44 Studierende waren, kann man vielleicht annehmen, das viele der 40 fehlenden Studierenden nur einen kleinen Teil der Vorlesung als neu empfinden.

Es war allerdings die letzte Vorlesung vor Weihnachten, und die Probeklausur war nicht angekündigt, das erklärt vielleicht auch die eher geringe Teilnahmequote.