Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Institut für Informatik
MLU Halle-Wittenberg

16. Februar 2007

Vorlesung “Objektorientierte Programmierung” — Klausur —

Name: ____________________

Matrikelnummer: ____________________

Studiengang: ____________________

Aufgabe	Punkte	von (Max.)	Zeit
1 (Arithmetischer Ausdruck, Zeiger)		1	3 min
2 (Logische Bedingung)		1	3 min
3 (Schleife)		1	3 min
4 (Array, C-String, Pointer)		1	3 min
5 (Globale und lokale Variablen)		1	3 min
6 (Rekursion)		1	3 min
7 (Klasse)		1	3 min
8 (Programm: Klasse für lange Zahlen)		10	30 min
9 (Fehlersuche)		3	5 min
10 (Verschiedene Fragen)		4	4 min
Zusatzaufgabe		0 (+1)	5 min
Summe		24	65 min

Hinweise:

- Bearbeitungsdauer: 90 Minuten
- Skript, Bücher, Notizen sind erlaubt. Notebooks, PDAs, etc. dürfen nicht verwendet werden. Mobiltelefone bitte ausschalten (oder mit Aufsicht besprechen).
- Die Klausur hat 13 Seiten. Bitte prüfen Sie die Vollständigkeit.
- Bitte benutzen Sie den vorgegebenen Platz. Wenn Sie auf die Rückseite ausweichen müssen, markieren Sie klar, daß es eine Fortsetzung gibt.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Fragen Sie, wenn Ihnen eine Aufgabe nicht klar ist!
- Zum (garantierten) Bestehen benötigen Sie 60% der Punkte (15/24).

Aufgabe 1 (Arithmetischer Ausdruck, Zeiger) 1 Punkt

Was gibt das folgende Programm aus?

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

int main()
{
        int i = 2;
        int j = 10;
        int* p = &j;

        j += 5;
        int n = i++ * *p + 4;
        cout << n;

        return 0;
}
```

Ausgabe: ____________________

Hier nochmal die Tabelle mit den Prioritätsstufen der Operatoren:

<table>
<tr><td>18</td><td>::</td><td>Gültigkeitsbereich</td></tr>
<tr><td>17</td><td>++ (Postfix), ., ->, [], f(), ...</td><td>Postfix-Operatoren</td></tr>
<tr><td>16</td><td>- (unär), !, * (deref), ++ (Präfix), ...</td><td>Präfix-Operatoren</td></tr>
<tr><td>15</td><td>.*, ->*</td><td>Zeiger auf Komp.</td></tr>
<tr><td>14</td><td>*, /, %</td><td>Multiplikation etc.</td></tr>
<tr><td>13</td><td>+, -</td><td>Addition, Subtraktion</td></tr>
<tr><td>12</td><td><<, >></td><td>Shift etc.</td></tr>
<tr><td>11</td><td><, <=, >, >=</td><td>kleiner etc.</td></tr>
<tr><td>10</td><td>==, !=</td><td>gleich, verschieden</td></tr>
<tr><td>9</td><td>&</td><td>Bit-und</td></tr>
<tr><td>8</td><td>^</td><td>Bit-xor</td></tr>
<tr><td>7</td><td>|</td><td>Bit-oder</td></tr>
<tr><td>6</td><td>&&</td><td>und</td></tr>
<tr><td>5</td><td>||</td><td>oder</td></tr>
<tr><td>4</td><td>?:</td><td>Bedingter Ausdruck</td></tr>
<tr><td>3</td><td>=, +=, -=, *=, /=, ...</td><td>Zuweisungen</td></tr>
<tr><td>2</td><td>throw</td><td>Exception auslösen</td></tr>
<tr><td>1</td><td>,</td><td>Sequenz</td></tr>
</table>

Bei gleicher Priorität sind alle Operatoren (außer Präfixoperatoren und Zuweisungen) implizit von links geklammert (linksassoziativ).

Aufgabe 2 (Logische Bedingung) 1 Punkt

Was gibt das folgende Programm aus?

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

int main()
{
        int i = 2;
        int j = 3;

        if(i != j && i > 5)
                cout << 10;
        else if(i > j || j <= 2)
                cout << 20;
        i++;
        if(i == j)
                cout << 30;
        else
                cout << 40;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 3 (Schleife) 1 Punkt

Was gibt das folgende Programm aus?

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

int main()
{
        int i = 2;
        int j = 3;

        while( i < 4) {
                i++;
                j++;
        }
        cout << i * j;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 4 (Array, C-String, Pointer) 1 Punkt

Was gibt dieses Programm aus?

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

int main()
{
        int i = 0;
        char s[5] = "abc";
        for(char* p = s; *p; p++)
                i++;
        cout << i;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 5 (Globale und lokale Variablen) 1 Punkt

Was gibt dieses Programm aus?

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

int n = 5;

int f(int i)
{
        i = i + n;
        return i;
}

int main()
{
        int n = 2;
        int j  = f(n);
        cout << n * 10 + j;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 6 (Rekursion) 1 Punkt

Was gibt dieses Programm aus?

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

int n = 0;

void f(int i)
{
        n++;
        if(i > 1)
                f(i/2);
        else
                cout << n << '.';
        n--;
}

int main()
{
        f(8);
        cout << n;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 7 (Klasse) 1 Punkt

Was gibt dieses Programm aus?

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

class C {
        int n;
    public:
        int f() { n = n * 10; }
        int g() { return n; }
        C(int i) { n = i; }
};

int main()
{
        C a(1);
        C* b = new C(2);
        a.f();
        a.f();
        b->f();
        cout << a.g() + b->g();

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 8 (Programm: Klasse für lange Zahlen) 10 Punkte

Definieren Sie eine Klasse `BigInt` für 100-stellige Zahlen, wobei Sie die Ziffern bitte in einem `int`-Array der Größe 100 abspeichern. Definieren Sie folgende Methoden für die Klasse:

- Konstruktor:
  Er soll die Zahl auf 0 initialisieren (alle Ziffern).
- `void set_digit(int i, int d)`:
  Diese Funktion soll die Ziffer `i` auf den Wert `d` setzen, sofern `i` zwischen `0` und `99` und `d` zwischen `0` und `9` liegt. Andernfalls soll die Methode "`Fehler!`" auf `cerr` ausgeben.
- `void print()`:
  Diese Methode soll den aktuellen Wert der Zahl ausgeben, und dabei die Ziffern von Position 99 (höchstwertige Ziffer) bis Position 0 (niedrigstwertige Ziffer) nacheinander auf `cout` ausgeben. Dabei sollen allerdings führende Nullen unterdrückt werden, d.h. es sollen nicht immer alle 100 Ziffern ausgegeben werden, sondern nur die Ziffern ab der maximalen Indexposition, die von 0 verschieden ist. Allerdings muß mindestens eine Ziffer ausgegeben werden, d.h. die Indexposition 0 muß in jedem Fall gedruckt werden.

Schreiben Sie dann bitte noch ein Hauptprogramm zum Testen. Im Hauptprogramm soll ein Objekt der Klasse angelegt werden, die Ziffer an der Stelle 3 auf 5 gesetzt werden, und die Zahl anschließend ausgegeben werden.

Versuchen Sie Syntaxfehler zu vermeiden und Ihr Programm leicht verständlich zu gestalten. Es können Punkte für schlechten Stil abgezogen werden. Es ist Platz für die Lösung dieser Aufgabe auf der nächsten Seite.

Platz für die Lösung von Aufgabe 8

Aufgabe 9 (Fehlersuche) 3 Punkte

Das folgende Programm enthält (mindestens) 5 Fehler. Bitte geben Sie drei dieser Fehler an (es ist möglicherweise schwierig, alle fünf zu entdecken). Falls Sie mehr als drei Fehler angeben, werden nur die ersten drei gewertet (es gibt keine Extrapunkte). Geben Sie bitte jeweils die Zeilennummer mit an, in der sich der Fehler befindet. Es zählt auch nicht als Fehler, daß das Programm nichts Sinnvolles tut. Direkte Folgefehler zählen auch nicht.

```
 (1)  #include <iostream>
 (2)  using namespace std;
 (3)
 (4)  class C {
 (5)  private:
 (6)          int a;
 (7)  public:
 (8)          int f() { return a; }
 (9)          C(int n) { a = n; }
(10)  }
(11)
(12)  int main()
(13)  {
(14)          C x;
(15)          int i;
(16)          x.a = 5;
(17)          int n = x;
(18)          cout << n + i;
(19)          return 0;
(20)  }
```

1. Zeile: ______

   Begründung: ________________________________

   ____

2. Zeile: ______

   Begründung: ________________________________

   ____

3. Zeile: ______

   Begründung: ________________________________

   ____

Aufgabe 10 (Verschiedene Fragen) 4 Punkte

Bitte kreuzen Sie jeweils die richtige Antwort an (es ist genau eine Antwort pro Teilaufgabe richtig). Wenn Sie nichts ankreuzen, ist der Punkt auf jeden Fall verloren. Sie sollten daher notfalls raten.

a) Was passiert, wenn in einer Klasse kein Konstruktor definiert wird?

- ☐ Der Compiler meldet einen Fehler.
- ☐ Der Compiler legt automatisch einen Konstruktor an, der alle Komponenten auf 0 initialisiert.
- ☐ Der Compiler legt automatisch einen an, der die Konstruktoren von objektwertigen Komponenten aufruft. Normale Komponenten (z.B. vom Typ `int`) werden nicht initialisiert.

b) Was ist der Unterschied zwischen `=` und `==`?

- ☐ `=` ist die Zuweisung, `==` der Vergleich.
- ☐ `=` vergleicht Strukturen Komponente für Komponente, `==` vergleicht die Adressen.
- ☐ `=` funktioniert für in C++ eingebaute Typen, `==` muß für selbstdefinierte Typen verwendet werden.

c) Was ist der Unteschied zwischen "call by value" und "call by reference"?

- ☐ Das ist nur ein internes Detail bei der Implementierung der Sprache, aus Programmierersicht macht es keinen Unterschied.
- ☐ Für rekursive Prozeduren kann man nur "call by value" verwenden. Ansonsten ist "call by reference" effizienter.
- ☐ Man muß "call by reference" verwenden, wenn man will, daß die aufgerufene Prozedur die übergebene Variable ändern kann.

d) Was bedeutet "`const char* p;`"?

- ☐ Das ist ein Syntaxfehler.
- ☐ Der Zeiger `p` kann verändert werden, aber der Speicherbereich, auf den `p` zeigt, wird nicht verändert.
- ☐ Der Zeiger `p` ist konstant.

Zusatzaufgabe (Vererbung) 1 Extrapunkt

Was gibt dieses Programm aus?

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

class C {
    public:
        int f() { return 1; }
        int g() { return 2; }
};

class D : public C {
    public:
        int g() { return 5; }
};

int main()
{
        D d;
        C* c = &d;
        cout << c->f() + c->g();

        return 0;
}
```

Ausgabe: ____________________