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

27. März 2007

Vorlesung "Objektorientierte Programmierung" — Nachholklausur —

Name: ____________________

Matrikelnummer: ____________________

Studiengang: ____________________

Aufgabe	Punkte	von (Max.)	Zeit
1 (Arithmetischer Ausdruck, Array)		1	3 min
2 (Logische Bedingung)		1	3 min
3 (Schleife)		1	3 min
4 (Array, Pointer)		1	3 min
5 (Globale vs. lokal, Parameterübergabe)		1	3 min
6 (Klasse)		1	3 min
7 (Referenzen)		1	3 min
8 (Programm: Primfaktorzerlegung)		7	20 min
9 (Programm: Klasse für Polynome)		7	20 min
10 (Fehlersuche)		3	5 min
Zusatzaufgabe		0	5 min
Summe		24	71 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 14 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!

Aufgabe 1 (Arithmetischer Ausdruck, Array) 1 Punkt

Was gibt das folgende Programm aus?

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

int f(int n)
{
        return n + 4;
}

int g(int n)
{
        n++;
        return n * n;
}

int main()
{
        int a[3];
        a[0] = 1;
        a[1] = 2;
        a[2] = 3;
        a[a[1]]++;
        cout << f(g(a[0]+a[2]));
        return 0;
}
```

Ausgabe: ____________________

Hier nochmal die Tabelle mit den Prioritätsstufen der Operatoren:

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

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

Aufgabe 2 (Logische Bedingung) 1 Punkt

Gegeben sei folgende Funktion:

```
int f(int a, bool b)
{
        if(b == true)
                a += 2;
        else if(!b || a > 0)
                a = 1;
        else
                a = 5;
        return a;
}
```

Welche der folgenden drei Funktionen liefert für alle Eingabewerte die gleiche Ausgabe, tut also exakt dasselbe wie die Funktion `f`?

- ☐
```
int f1(int a, bool b)
{
        if(b)
                return a + 2;
        return 1;
}
```

- ☐
```
int f2(int a, bool b)
{
        if(b == false && a <= 0)
                a = 5;
        if(b == false && a > 0)
                a = 1;
        a += 2;
        return a;
}
```

- ☐
```
int f3(int a, bool b)
{
        if(b) {
                a++;
                a++;
        }
        else {
                if(a > 0)
                        a++;
                else
                        a += 5;
        }
        return a;
}
```


Aufgabe 3 (Schleife) 1 Punkt

Was gibt das folgende Programm aus?

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

int main()
{
        int n = 0;

        for(int i = 1; i < 10; i *= 2) {
                n++;
        }
        cout << n;

        return 0;
}
```

Ausgabe: ____________________________

Aufgabe 4 (Array, Pointer) 1 Punkt

Was gibt dieses Programm aus?

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

const int n = 5;

int main()
{
        int  a[n];
        int* p = a;

        for(int i = 0; i < n; i++)
                a[i] = i * 10;

        *p += 1;
        p  += 2;
        *p += 3;
        cout << *p;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 5 (Global vs. lokal, Parameterübergabe) 1 Punkt

Was gibt dieses Programm aus?

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

int m = 1;
int n = 3;

void f(int n)
{
        n++;
        m = n + 5;
}

void g(int* j)
{
        j += n;
}

int main()
{
        int n = 4;
        f(n);
        g(&n);
        cout << n * m;

        return 0;
}
```

Ausgabe: ______________________________

Aufgabe 6 (Klasse) 1 Punkt

Was gibt dieses Programm aus?

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

class C {
        static int next;
        int id;
    public:
        int get_id() const { return id; }
        C() { id = next++; }
};

int C::next = 1;

int main()
{
        C a;
        C* b = new C;
        cout << b->get_id();

        return 0;
}
```

Ausgabe: ________________________________

Aufgabe 7 (Referenzen) 1 Punkt

Was gibt dieses Programm aus?

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

int f(int& n)
{
        return n * n;
}

int main()
{
        int m = 1;
        int& r = m;
        r++;
        int a = f(m);
        cout << a * 10 + m;

        return 0;
}
```

Ausgabe: ________________________________

Aufgabe 8 (Programm: Primfaktorzerlegung) 7 Punkte

Schreiben Sie ein Programm, das eine positive ganze Zahl `n` einliest, und die Primfaktoren von `n` nach folgendem, sehr einfachen Verfahren berechnet:

- Beginnend mit `i = 2` wird nacheinander für immer größere Zahlen `i` getestet, ob der aktuelle Wert von `n` durch `i` teilbar ist (zur Erinnerung: `n % i` ist der Divisionsrest bei Teilung von `n` durch `i`).
- Falls das der Fall ist (`n` ist durch `i` teilbar), wird `i` ausgegeben (es ist dann ein Primfaktor), und `n` wird auf `n / i` gesetzt. Bitte geben Sie nach jedem Primfaktor einen Zeilenumbruch aus. Anschließend muß dann noch einmal für den gleichen Wert von `i` geprüft werden, ob `n` durch `i` teilbar ist (der Primfaktor könnte ja mehrfach in `n` enthalten sein und soll dann auch mehrfach ausgegeben werden).
- Falls `n` nicht (mehr) durch `i` teilbar ist, wird `i` um `1` erhöht. Die Schleife endet, wenn `i` größer als `n` ist (`n` hat in diesem Fall den Wert `1`).

Beispiel:

- Angenommen, man gibt den Wert `75` für `n` ein.
- Zuerst wird geprüft, ob `n` durch `2` teilbar ist. Das ist nicht der Fall.
- Als nächstes wird geprüft, ob `n` durch `3` teilbar ist. Das trifft zu, also wird `3` ausgegeben, und `n` auf `n/3`, d.h. `25` gesetzt.
- Nun wird noch einmal geprüft, ob `n` durch `3` teilbar ist. Das ist nicht der Fall (`n` ist jetzt ja `25`), also wird der Teiler wieder erhöht.
- Als nächstes wird also geprüft, ob `n` durch `4` teilbar ist. Das ist eigentlich überflüssig, da es ja schon nicht durch `2` teilbar ist, aber dieses Verfahren ist eben sehr einfach. Das Ergebnis ist natürlich negativ, also wird der Teiler wieder erhöht.
- `25` ist durch `5` teilbar, also wird `5` ausgegeben, und `n` auf `n/5`, also `5` gesetzt.
- Weil der letzte Test erfolgreich war, wird der Teiler `5` noch einmal geprüft. Das Ergebnis ist wieder positiv, also wird `5` noch einmal ausgegeben, und `n` auf `n/5`, d.h. `1`, gesetzt.
- Nun ist der aktuelle Teiler (immernoch `5`) größer als der aktuelle Wert von `n`, also endet die Schleife.

Geben Sie das vollständige Programm an und versuchen Sie Syntaxfehler zu vermeiden. Gestalten Sie Ihr Programm möglichst leicht verständlich und vermeiden Sie unnötige Komplikationen. Es können Punkte für schlechten Stil abgezogen werden.

Bitte geben Sie für `n` einen Eingabeaufforderung aus, z.B. “`Bitte positive ganze Zahl eingeben:`”. Sie brauchen keine ungültigen Eingaben zu behandeln, d.h. Sie können sich darauf verlassen, daß dann wirklich eine positive ganze Zahl eingegeben wird.

Es ist Platz für die Lösung dieser Aufgabe auf der nächsten Seite.

Platz für die Lösung von Aufgabe 8

Aufgabe 9 (Programm: Klasse für Polynome) 7 Punkte

Definieren Sie eine Klasse `Poly2` für Polynome vom Grad 2, also Funktionen der Form

$$f(x) = a * x^2 + b * x + c$$

D.h. ein Objekt dieser Klasse soll die Koeffizienten a, b, c enthalten (jeweils als `double`-Werte). Diese Koeffizienten sollen beim Aufruf des Konstruktors angegeben werden, und von außen nur lesend über Methoden `get_a`, `get_b`, `get_c` zugreifbar sein. Außerdem soll es eine Methode `eval` geben, die den Funktionswert $f(x)$ an der Stelle x berechnet. Die Klasse soll also folgende Methoden haben:

- Konstruktor (mit Parametern `a`, `b`, `c` vom Typ `double`).
- `get_a` (ohne Parameter): Liefert den Wert von `a`, also ein `double`).
  `get_b`, `get_c` entsprechend.
- `double eval(double x)`: Liefert $a * x^2 + b * x + c$.

Schreiben Sie dann bitte noch ein Hauptprogramm zum Testen. Im Hauptprogramm soll ein Objekt der Klasse angelegt werden, die drei Koeffizienten abgefragt und ausgegeben werden, sowie ein Funktionswert berechnet und ausgegeben werden. Die Ausgabe des Testprogramms sollte ungefähr so aussehen:

```
a = 1
b = 2
c = 3
f(5) = 38
```

Benutzen Sie bitte die hier angegebenen Werte für `a`, `b`, `c` und `x` zum Test.

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 9

Aufgabe 10 (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 nicht als Fehler, daß das Programm nichts Sinnvolles tut. Direkte Folgefehler zählen auch nicht.

```
 (1)  #include <iostream>
 (2)  using namespace std;
 (3)
 (4)  int f(int n)
 (5)  {
 (6)      int m = f(n-1);
 (7)      return m * m;
 (8)  }
 (9)  void g(int n)
(10)  {
(11)      return n * 5;
(12)  }
(13)  int main()
(14)  {
(15)          C x;
(16)          int a[5];
(17)          a[5] = 27;
(18)          a[0] = f(0);
(19)          g(3);
(20)          delete x;
(21)          return 0;
(22)  }
```

1. Zeile: ______

   Begründung: ______________________________

   ____

2. Zeile: ______

   Begründung: ______________________________

   ____

3. Zeile: ______

   Begründung: ______________________________

   ____

Zusatzaufgabe (Vererbung) 1 Extrapunkt

Was gibt dieses Programm aus?

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

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

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

class E : public D {
    public:
        int f() { return 13; }
};

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

        return 0;
}
```

Ausgabe: ______________________________