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

22. März 2011

Vorlesung "Objektorientierte Programmierung"
— Klausur —

Name: _____

Matrikelnummer: _____

Studiengang: _____

Aufgabe	Punkte	von	Zeit
1 (Programmierung: Klasse)		30+2	30 min
2 (Variablenwerte, Pointer, Referenzen)		8	10 min
3 (Logische Bedingungen)		6	10 min
4 (Typfehler)		10	10 min
5 (Fehlersuche, Klassen)		6	10 min
6 (Vererbung)		10	10 min
7 (Array, C-String)		0+2	10 min
Summe		70+4	90 min

☐ Ich fühle mich gesundheitlich in der Lage, diese Prüfung abzulegen. (Bitte sprechen Sie mit dem Aufsichtspersonal, falls Sie sich krank fühlen.)

☐ Für den Fall, daß ich nicht korrekt zu dieser Prüfung angemeldet sein sollte, melde ich mich hiermit unwiderruflich an. Sollte der Prüfungsausschuss die verspätete Anmeldung nicht akzeptieren, erkläre ich mich auch damit einverstanden, dass die Klausur nicht gewertet wird.

Unterschrift: _____

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 (42/70). Die Grenze wird möglicherweise gesenkt.

Zum Nachschlagen:

- 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).

- Bei der Integer-Division ist für positive Eingabewerte garantiert, daß abgerundet wird, z.B. liefert `8/3` das Ergebnis `2`.

- Der Divisionsrest (Modulo) Operator ist `%`.

Aufgabe 1 (Programmierung: Klasse) 30 Punkte

Schreiben Sie in C++ eine Klasse `Tagesplaner`, mit welchem Sie die 24 Stunden eines Tages ähnlich einem Kalender verwalten können. In diesem Tagesplaner können Sie Zeiträume für Veranstaltungen eintragen. Diese müssen jedoch immer zu einer vollen Stunde beginnen und auch die Dauer der Veranstaltung muss in ganzen Stunden angegeben werden. Im Gegensatz zum realen Tagesplaner soll in unserer Klasse `Tagesplaner` nur vermerkt werden, von wann bis wann eine Veranstaltung stattfindet (welche Stunden also schon belegt sind). Die Veranstaltungsnamen werden vernachlässigt und sollen nicht gespeichert werden. Beispiel:

Realer Tagesplaner

0
1
2
3
4
5
6
7
8:00 Uhr – 10:00 Uhr Vorlesung
10
11
12
13
14
15
16:00 Uhr – 18:00 Uhr Frisör
18
19
20:00 Uhr – 23:00 Uhr Kino
23

Array `Stunden` in der Klasse `Tagesplaner`

0	false
1	false
2	false
3	false
4	false
5	false
6	false
7	false
8	true
9	true
10	false
11	false
12	false
13	false
14	false
15	false
16	true
17	true
18	false
19	false
20	true
21	true
22	true
23	false

Sie sind in der Implementierung der Klasse frei: Nur die geforderten Methoden müssen so funktionieren, wie auf der nächsten Seite beschrieben. Es bietet sich natürlich an, intern in der Klasse ein Array der Größe 24 zu verwenden. Ein Array von booleschen Werten würde ausreichen. Im obigen Beispiel steht `true` in einem Feld, wenn eine Stunde belegt ist und somit zu einer Veranstaltung gehört. Ist eine Stunde noch frei, so wird dies durch `false` gekennzeichnet.

Implementieren Sie folgende Methoden:

- Einen Konstruktor, welcher das Array initialisiert.
- Methode `Termin_eintragen`: Diese bekommt als ersten Parameter die Startzeit eines neuen Termins und im zweiten Parameter die Dauer (in ganzen Stunden) übergeben. Die Startzeit muss zwischen 0 und 23 liegen, die Dauer ist mindestens 1, maximal ist Startzeit plus Dauer 24. Sie können also davon ausgehen, dass keine Veranstaltung über Mitternacht hinaus geht.

  Die Methode soll `true` zurückliefern, falls über diesen Zeitraum noch keine andere Veranstaltung geplant ist. In diesem Fall soll der neue Termin im Array eingetragen werden. Gibt es Überschneidungen mit bereits eingetragenen Veranstaltungen, so gibt die Methode `false` zurück. Das Array soll dann unverändert bleiben, d.h. der Termin soll nicht (auch nicht teilweise) eingetragen werden.

  Angenommen, man hat ein Objekt `MeinTag` der Klasse `Tagesplaner` angelegt. Im Beispiel würde der Aufruf zum obigen Frisörtermin dann so aussehen:

  ```
  bool ok = MeinTag.Termin_eintragen(16,2);
  ```

  Implementierungs-Hinweis: Es bietet sich an, als erstes mit einer Schleife zu prüfen, ob der Zeitraum noch frei ist. Falls das so ist, kann man dann im Anschluss in einer separaten Schleife den Termins eintragen.

  Extra-Punkte: Sie müssen die Parameter-Werte nicht prüfen. Falls Sie das aber tun, und im Fehlerfall `false` zurückgeben, bekommen Sie zwei Extra-Punkte.

- Eine Methode `Freizeit`, welche alle freien Zeiträume bestimmt und ausgibt. Diese Methode hat keine Parameter und keinen Rückgabewert. Es soll pro Zeile ein Zeitraum in der Form `Startzeit - Endzeit` ausgegeben werden. Die Ausgabe der Methode zum obigen Beispiel lautet somit:

  ```
  0:00 - 8:00
  10:00 - 16:00
  18:00 - 20:00
  23:00 - 24:00
  ```

  Implementierungs-Hinweis: Natürlich verwendet man auch hier eine Schleife über das Array. Interessant sind dann die Wechsel zwischen belegten und unbelegten Stunden. Bei einem Wechsel von belegt nach unbelegt hat man den Beginn eines Freizeit-Bereiches, beim Wechsel von unbelegt zu belegt das Ende. Diese Wechsel kann man verschieden erkennen, z.B. mit einer booleschen Variable, oder indem man jeweils zwei aufeinanderfolgende Indexpositionen vergleicht. Kritisch sind dann der Beginn und das Ende. Achten Sie darauf, dass Sie auch Freizeit, die bei 0 Uhr beginnt, oder bei 24 Uhr endet, behandeln können.

Hinweise: Es ist Platz für die Lösung auf dieser und der nächsten Seite. Beachten Sie, daß auch für ein fehlendes oder falsches Semikolon ein Punkt abgezogen werden kann. Versuchen Sie also, Syntaxfehler zu vermeiden. Auch ein fehlender Zeilenvorschub in den Ausgabeanweisungen kann Sie einen Punkt kosten. Bemühen Sie sich außerdem um Verständlichkeit und guten Programmierstil. Es können auch für schlechten Stil Punkte abgezogen werden! Sie brauchen aber nicht jede einzelne Anweisung zu kommentieren. Entscheidend ist, dass unsere Korrekteure Ihr Programm zügig und ohne Probleme verstehen können.

Platz für die Losung von Aufgabe 1 (Klasse `Tagesplaner`), Forts.:

Aufgabe 2 (Variablenwerte, Pointer, Referenzen) 8 Punkte

Füllen Sie im folgenden Programmausschnitt die beiden Tabellen mit den Werten der Variablen zum jeweiligen Zeitpunkt aus, wenn die Funktion `f` mit `f(2)` aufgerufen wird. Wenn eine Variable nicht initialisiert sein sollte, schreiben Sie "nicht initialisiert", bei Pointern und Referenzen geben Sie bitte an, auf welche Variable der Pointer/die Referenz zeigt. Es gibt einen Punkt pro korrekter Antwort.

```
void g(int n)
{
    n = 3;
}

void h(int &n)
{
    n = 4;
}

int f(int n)
{
    int i;
    int j = 0;
    int *p;
    p = &i;
```

n	
i	
j	
p	

```
    g(n);
    h(j);
    *p = n * 3;
```

n	
i	
j	
p	

```
    return i;
}
```

Aufgabe 3 (Logische Bedingungen) 6 Punkte

Kreuzen Sie in folgendem Programm die Ausgabeanweisungen an, die mindestens einmal ausgeführt werden (also tatsächlich etwas ausgeben). Pro korrekte Antwort (angekreuzt/nicht angekreuzt) gibt es einen Punkt.

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

int main()
{
    int a = 1;
    int b = 2;
    int c = 3;
    bool t = (a < b);
    if(t && a != 1)
        cout << 1;          ⟵ ☐ wird ausgeführt
    if(!t || b == 2) {
        cout << 2;          ⟵ ☐ wird ausgeführt
        if(c > 1 && c <= 10)
            cout << 3;      ⟵ ☐ wird ausgeführt
    }
    if(c)
        cout << 4;          ⟵ ☐ wird ausgeführt
    while(c % 2 == 0) {
        cout << 5;          ⟵ ☐ wird ausgeführt
        c++;
    }
    if(c = 1) // Achtung! = statt ==
            cout << 6;      ⟵ ☐ wird ausgeführt
}
```

Aufgabe 4 (Typfehler) 10 Punkte

Welche der folgenden Zuweisungen ergeben einen Typfehler, d.h. sind in C++ unzulässig? Spielen Sie also Compiler und kreuzen Sie die Anweisungen an, für die Sie eine Fehlermeldung ausgeben würden. Bei den korrekten (in C++ erlaubten) Anweisungen kreuzen Sie nichts an. Pro richtige Antwort (angekreuzt/nicht angekreuzt) gibt es einen Punkt.

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

int main()
{
    char a[5];
    char *p;
    int i = 2;
    int *q = 0;

     a[0] = 0;   ☐ falsch (Typfehler)

     p = a;      ☐ falsch (Typfehler)

     p = a[1];   ☐ falsch (Typfehler)

     p = 'a';    ☐ falsch (Typfehler)

     i = 5;      ☐ falsch (Typfehler)

     i = 'a';    ☐ falsch (Typfehler)

     i = "a";    ☐ falsch (Typfehler)

     p = a+1;    ☐ falsch (Typfehler)

     p = q;      ☐ falsch (Typfehler)

     q = 5;      ☐ falsch (Typfehler)

}
```

Aufgabe 5 (Fehlersuche, Klassen) 6 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. Fehler sind Syntaxfehler, Typfehler, Berechtigungsfehler (alles, was der Compiler als Fehler melden würde), aber keine inhaltlichen Fehler (z.B., dass das Programm nichts Sinnvolles tut). Direkte Folgefehler zählen nicht. Es ist Platz für die Antwort auf der nächsten Seite.

```
 (1)  #include <iostream>
 (2)  using namespace std;
 (3)
 (4)  class C {
 (5)          int f(int n)
 (6)                  { return n * a; }
 (7)  public:
 (8)          void g(int n)
 (9)                  { a = f(n+2);
(10)                    return n % 3;
(11)                  }
(12)          C(int n);
(13)  private:
(14)          int a;
(15)  };
(16)
(17)  C::C(int n)
(18)  {
(19)          a = n;
(20)  }
(21)
(22)  int main()
(23)  {
(24)          C x;
(25)          int n = 10;
(26)          x->g(2);
(27)          if(n > 2) {
(28)                  C *y = new C(3);
(29)                  y->g(4);
(30)          }
(31)          y->g(5);
(32)          cout << x.f(2);
(33)          return 0;
(34)  }
```

1. Zeile: _____

   Begründung: _____

   _____

2. Zeile: _____

   Begründung: _____

   _____

3. Zeile: _____

   Begründung: _____

   _____

Aufgabe 6 (Vererbung) 10 Punkte

Was gibt dieses Programm aus? (Es gibt einen Punkt pro korrekter Antwort.)

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

class C {
    public:
        int a;
        C() { a = 10; }
        int f() { return 20; }
        int g() { return 30; }
        virtual int h() { return 40; }
};

class D : public C {
    public:
        D() { a = 15; }
        int g() { return 35; }
        int h() { return 45; }
};

int main()
{
        C *c = new C();
        D *d = new D();
        C *x = d;

        cout << c->f(); _____

        cout << c->g(); _____

        cout << c->h(); _____

        cout << d->f(); _____

        cout << d->g(); _____

        cout << d->h(); _____

        cout << x->f(); _____

        cout << x->g(); _____

        cout << x->h(); _____

        cout << x->a;   _____

        return 0;
}
```

Aufgabe 7 (Array) 2 Extrapunkte

Die Methode `copy` in der folgenden Klasse `C` enthält (mindestens) einen Programmierfehler, der aber kein Syntaxfehler ist, d.h. der Compiler meldet keinen Fehler. Dennoch wird diese Methode bei bestimmten Parameterwerten nicht so funktionieren, wie der Programmierer vermutlich beabsichtigt hat. Was kann passieren? Geben Sie ein Beispiel für einen problematischen Aufruf und erklären Sie, wie sich das Programm dann verhält.

Da es für diese Aufgabe Zusatzpunkte gibt, wird sie relativ streng bewertet. Sie erhalten die Zusatzpunkte nur, wenn Sie ein ernstes Problem zutreffend beschreiben.

```
(1)  const int maxsize = 5;
(2)
(3)  class C {
(4)  private:
(5)          char a[maxsize];
(6)  public:
(7)          C() {
(8)                  a[0] = '\0';
(9)          }
(10)         void copy(const char *s) {
(11)                 int i = 0;
(12)                 while(*s) {
(13)                         if(i < maxsize) {
(14)                                 a[i++] = *s;
(15)                                 s++;
(16)                         }
(17)                 }
(18)                 a[i] = '\0';
(19)         }
(20) };
```

Problematischer Aufruf (für Objekt `x` der Klasse `C`):

Verhalten: