1. Klausur

zur Vorlesung

„Objektorientierte Programmierung in C++“

(Abschluß 1. Semester) 1998/99

Bitte notieren Sie auf allen (!) Lösungsblättern Ihren Namen und ihre Matrikelnummer. Die Rückgabe der Ergebnisse und Diskussion von Beispiellösungen erfolgt am 16.2.98. Mit der vollständigen Lösung der folgenden 8(+1) Aufgaben können insgesamt 76(+6) Punkte erreicht werden. Streben Sie bei der Auswahl der von Ihnen gelösten Aufgaben 24 Punkte an!
Viel Erfolg !!!

---

1. Easthereggs [8 Punkte]

Auf den jungen *Carl Friedrich Gauß* geht der folgende Algorithmus zur Berechnung des Osterdatums (Ostersonntag) zurück [Quelle: "Weser-Kurier" Bremen vom 18.04.1991]:

Ostern ist am (22+d+e).März oder am (d+e-9).April, je nachdem,
ob (22+d+e) < 32 ist oder nicht. Die Berechnung von d und e ist leider
etwas aufwendig, und es gilt:

j = 100p+n	Keine Regel ohne Ausnahme:
q = ganzzahliger Teil von p/3	Ist d = 29 und e = 6, so ist Ostern am 19.April.
r = ganzzahliger Teil von p/4	Ist d = 28 und e = 6, so ist Ostern am 18.April.
x = Dreißigerrest von 15+p-q-r	
y = Siebenerrest von p-r+4	Beispiel für Ostern 1999:
a = Neunzehnerrest von j	j = 1999 p = 19, n = 99, q = 6, r = 4, x = 24,
b = Viererrest von j	y = 5, a = 4, b = 3, c = 4, d = 10, e = 3,
c = Siebenerrest von j	22+d+e=35 > 32, d+e-9 = 4
d = Dreißigerrest von 19a+x	Also 4. April
e = Siebenerrest von 2b+4c+6d+y	

Implementieren Sie ein C++ -Programm, welches nach diesem Algorithmus die Osterdaten der Jahre 1900 bis 2100 ermittelt und ausgibt. Benutzen Sie eine Klasse **`Easter`** mit (mindestens) einer Methode **`void print();`**, die das errechnete Datum (nach **`cout`**) ausgibt und mit einem Konstruktor, der die Jahreszahl übernimmt. Versuchen Sie, den gegebenen Algorithmus strukturell identisch umzusetzen (incl. aller Zwischenwerte), dabei aber dennoch möglichst effizient vorzugehen!

---

2. € to DM to € to DM .... [10 Punkte]

Schreiben Sie ein C++ -Programm, welches die Umrechnung zwischen Euro und DM vornimmt. In Abhängigkeit davon, wie es aufgerufen wird, soll dasselbe Programm beide Umrechnungen vornehmen können:

```
usage:
        euro      amount - calculates to DM
        dem       amount - calculates to Euro
```

Der umzurechnende Betrag soll jeweils als (einziger) Programmparameter eingegeben werden. Es ist zu prüfen, ob ein korrekter Programmaufruf (d.h. mit Parameter) vorliegt. Ansonsten ist eine geeignete **`usage`**-Ausschrift (s.o.) auszugeben. [**Hinweis:** Sie können davon ausgehen, dass der übergebene Programmparameter (wenn er vorliegt) richtig formatiert ist (Trennzeichen: Punkt, Mantisse zweistellig) und dessen Wert mit Hilfe von **`double atof(const char*);`** aus dem String in eine **`double`**-Zahl umwandeln lassen. Die Resultate der Umrechnung sollen jeweils auf ganze Pfennige/Cent gerundet werden!

Verwenden Sie die beiden Klassen:

```
class DM {
  ....
public:
  DM(int m, int p):mark(m), pfennig(p){}
  ...
  Euro toEuro();
  void out();
};

class Euro {
  ...
public:
  Euro(int e, int c):euro(e), cent(c){}
  ...
  DM toDM(); // 1 € == 1.95583 DM
  void out();
};
```

---

3. Statistik für Anfänger [10 Punkte]

Schreiben Sie eine C++ -Klasse, mit der (potentiell) beliebig große reellwertige Stichproben erfasst und statistisch ausgewertet werden können. Es gelte:

$$\mu[\text{mean}] = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \delta[\text{deviation}] = \sqrt{\sum_{i=1}^{n} (x_i - \mu)^2}$$

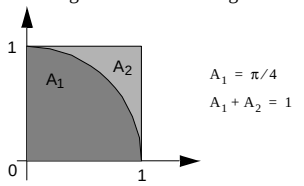
[**Hinweis:** Erfassen Sie alle Messwerte in einem **`double`**-Feld, welches ggf. zur Laufzeit vergrößert wird !]

```
class Stat {
  ...
public:
  Stat();
  ~Stat();
  void sample (const double);     // Stichprobenwert erfassen
  double min();                   // Minimaler Stichprobenwert
  double max();                   // Maximaler Stichprobenwert
  double mean();                  // Mittelwert der Stichprobe
  double deviation();             // Standardabweichung der Stichprobe
};
```

---

4. π [6 Punkte]

Bestimmen Sie mit einem C++ -Programm einen Näherungswert für π. Tun Sie dies, indem Sie von zufällig gewählten Punkten im Quadrat x=[0...1], y=[0...1] diejenigen zählen, die innerhalb des Einheitskreises liegen und diese Zahl in Relation zur Gesamtzahl n der Punkte für n=100, n=10000 und n=1000000 setzen. [**Hinweis:** Die Funktion **`double drand48();`** liefert Werte einer reellwertigen Gleichverteilung im Intervall [0,1) .]



$A_1 = \pi/4$

$A_1 + A_2 = 1$

5. Key Concepts [10 Punkte]

Welche Ausgaben produziert das folgende korrekte C++ Programm ?

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

inline void o(int i)              {cout<<i;}
inline void o(char c='-')         {cout<<c;}

class A {
public:
   A()                            {o(1);}
   void foo()                     {o(2);}
   virtual void bar()             {o(3);}
} a;

class B: public A {
public:
   B(int=0)                       {o(4);}
   B(const B&)                    {o(5);}
   void foo(int=0)                {o(6);}
   virtual void bar(int=1)        {o(5);}
};

class C: public B {
   A a();
   B b;
public:
   C(int i):b(0)                  {o(4);}
   C(const C&)                    {o(3);}
   virtual void foo(B=0)          {o(2);}
   virtual void bar()             {o(1);}
};

#define DO(X) X.foo(); p=&X; p->bar();
int main() {
   B b(9);
   C& c=*new C(8);
   o();
   A*p=0;
   DO(a);     DO(b);     DO(c);
   o('\n');
}
```

[Hinweis: Die Ausgabe dieses Programmes ist zugleich die *decryption phrase* für die Entschlüsselung der vorbereiteten Beispiellösungen unter

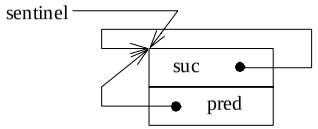
```
ftp://ftp.informatik.hu-berlin.de/pub/local/vorlesung/c++/1.klausur.99/loesungen.asc
```

Auspacken mittels:

```
pgp loesungen.asc
[Enter pass phrase: ....... ]
[Should 'loesungen' be renamed to 'loesungen.gz' (Y/n)? n ]
tar xvzf loesungen
]
```

6. Sentinel [12 Punkte]

Implementieren Sie eine C++ -Klasse **L**, die all ihre Objekte (und alle Objekte ihrer Ableitungen) in einer doppelt verketteten Liste erfasst. Jedes **L**-Objekt soll bei seiner Konstruktion einen Namen in Form einer Zeichenkette erhalten. Sofern ein **L**-Objekt als Kopie eines anderen entsteht, soll dessen Name übernommen und mit einem bei 0 beginnenden Zähler verlängert werden (maximal zweistellig). [**Hinweis:** Die Behandlung doppelt verketteter Listen vereinfacht sich enorm, wenn man die leere Liste durch ein künstlich eingeführtes **L**-Objekt als Wächter (*Sentinel*) realisiert: beim Einfügen braucht die leere Liste nicht als Sonderfall betrachtet werden; die Liste ist de facto leer, wenn bei dem über den Listenzeiger erreichten Element gilt: Vorgänger == Nachfolger==dieses Element]

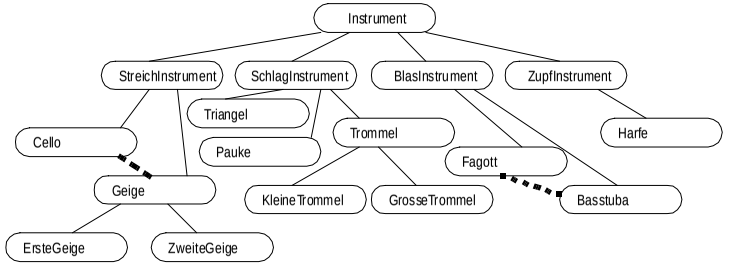



```
class L {
   char* const name;
   static L* const sentinel;
   static L s;
   ...
public:
   L(const char* n);
   ~L();
   static void out();
};
...
L L::s; L* L::sentinel=&s;
```

Durch Aufruf der Funktion **`L::out();`** soll es in einem Programm möglich sein, die im Moment des Aufrufs existierenden **L**-Objekte (incl. aller Objekte von Ableitungen und aller Kopien) namentlich aufzulisten.

7. Concerto grosso [12 Punkte]

Realisieren Sie die folgende Klassenhierarchie



und benutzen Sie diese in einem Programm, um ein Orchester einzurichten, welches aus

- 12 ersten Geigen und 10 zweiten Geigen,
- je 8 Violoncelli und Bratschen,
- 6 Kontrabässen,
- je 3 Fagotten, Flöten und Klarinetten,
- 4 Hörnern,
- je 3 Trompeten und Posaunen,
- je 2 Pauken, grossen Trommeln und kleinen Trommeln,

- sowie einer Basstuba einer Harfe und einem Triangel

besteht. Allen Instrumenten sind folgende Operationen eigen: **`void stimme();`** und **`void spiele();`** Schlag- und Blasinstrumente braucht man nicht stimmen. Streich- und Zupfinstrumente werden (symbolisch) jeweils auf die gleiche Weise gestimmt:

```
StreichInstrument::stimme()->    cout<<"ein Streichinstrument wird gestimmt\n";
```

Jedes Instrument soll beim Spielen (symbolisch) seinen eigenen Klang entfalten z.B.:

```
ErsteGeige::spiele()      ->             cout<<"es spielt eine Erste Geige\n";
```

Nach der Einrichtung des Orchesters:

```
Instrument* Orchester[genuegend_viele]; // z.B. 100
int n=0;
// 12 erste Geigen:
for (int i=0; i<12; ++i) Orchester[n++]=new ErsteGeige;
... // weitere Instrumente ´erzeugen´
Orchester[n++]=new Triangel;

cout<< n << " Instrumente im Orchester\n";
```

soll gestimmt und dann gespielt werden:

```
for (int i=0; i<n; ++i) Orchester[i]->stimme();
for (int i=0; i<n; ++i) Orchester[i]->spiele();
```

---

8. Fehlertext [8 Punkte]

Finden Sie möglichst viele Fehler in dem folgenden vermeintlichen C++-Programm, welches dazu dienen soll, die Längen von CD-Tracks (in Minuten:Sekunden) aufzuaddieren, um abschließend die Gesamtlänge aller Tracks in Minuten (mit Kommastellen) auszugeben.

```
/*   1 */ # include iostream
/*   2 */ # include <cassert>
/*   3 */
/*   4 */ //* track timer:*//
/*   5 */ class TT {
/*   6 */         int hour; min; sec;
/*   7 */         TT : min(0), sec(0){};
/*   8 */         TT add (int delay=0, int m; int s);
/*   9 */ double length() // in minutes       {
/*  10 */         retrun hour*60+min+sec/60;
/*  11 */         }
/*  12 */ }
/*  13 */
/*  14 */ TT:TT add (int delay=0, int m; int s) {
/*  15 */         assert (s<60);
/*  16 */         sec+=delay+s;
/*  17 */         if (s>=60) sec-=60; ++min;
/*  18 */         min=+m;
/*  19 */         if (m>=60) min-=60; ++hour;
/*  20 */         return this
/*  21 */ }
/*  22 */
/*  23 */ void main()
/*  24 */ {
/*  25 */    {
/*  26 */         // Dire Straits: Money for Nothing
/*  27 */         TT t();
/*  28 */         t.add(5,46)     // Sultans of Swing
/*  29 */         .add(4,00)      // Down to the Waterline
/*  30 */         .add(4,33)      // Portobello Belle
/*  31 */         .add(3,30);     // Twisting by the Pool
/*  32 */         .add(8,09);     // Tunnel of Love
/*  33 */         .add(5,57);     // Romeo and Juliet
/*  34 */         .add(3,31);     // Where do you think you're going
/*  35 */         .add(4,07);     // Walk of Life
/*  36 */         .add(5,48);     // Private Investigations
/*  37 */         .add(12,00);    // Telegraph Road
/*  38 */         .add(4,05);     // Money for Nothing
/*  39 */         .add(4:48);     // Brothers in Arms
/*  40 */
/*  41 */         cout<<"Gesamtlaenge: << t -> length << minuten"<<endl;
/*  42 */ }
```

(*) 9. Zugabe [6 Punkte]

Welches Problem verbirgt sich in dem folgenden C++ -Programm ? Es lässt sich fehlerfrei übersetzen, stürzt aber zur Laufzeit mit einer *segmentation violation* ab !

```
// Copyright © 1997, Gimpel Software
// bug1547.cpp: „Fehler des Monats“ bei www.gimpel.com

class Object { public: virtual double Evaluate() {return 0;} };
class AnimalObject : public Object {
   int someAnimalProperty;
public:
   double Evaluate(){return 1;}
};

double sum( Object ob[], int n )
{
   double r = 0;
   for( int i = 0; i < n; i++ )
        r += ob[i].Evaluate();
   return r;
}

int main() {
   AnimalObject aoa[1000];           /* animal object array */
   double aoa_s = sum( aoa, 1000 );  /* the sum of its members */
}
```