Abschlußklausur

zur Vorlesung

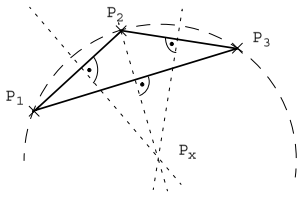
„Objektorientierte Programmierung in C++“

(Abschluß 2. Semester) 1997

Bitte notieren Sie auf allen Lösungsblättern Ihren Namen und einmalig Ihre Einschreibnummer. Anhand der letzteren werden Anfang August die Ergebnisse der Klausur durch Aushang (Axel-Springer-Str. 54a, 1. Etage rechts, Schaukasten) bekanntgegeben. Alle dort entsprechend markierten Studenten sind aufgefordert, sich für eine Prüfungszulassung zu zusätzlichen Konsultationen anzumelden (Dr. Ahrens, Tel. 20 181 238).
Mit der vollständigen Lösung der folgenden 8(+1) Aufgaben können insgesamt 105(+5) Punkte erreicht werden. Streben Sie bei der Auswahl der von Ihnen gelösten Aufgaben 40 Punkte an !

1. *Umkreiskonstruktion* **[20 Punkte]**

Füllen Sie die folgenden Klassenfragmente so aus, daß die entstehenden Klassen auf einfache Weise die konstruktive Bestimmung des Mittelpunktes P_x des Umkreises zu drei vorgegebenen Punkten $P_1=(x_1,y_2)$, $P_2=(x_2,y_2)$ und $P_3=(x_3,y_3)$ gestattet (Schnittpunkt der Mittelsenkrechten der Strecken $\overline{P_1P_2}$, $\overline{P_2P_3}$ und $\overline{P_3P_1}$). Geben Sie ein Programm an, das entsprechend der geometrischen Konstruktion arbeitet und anschließend die Korrektheit der Lösung überprüft. (P_x.Abstand(P_1)=P_x.Abstand(P_2)=P_x.Abstand(p_3))




```
class Punkt {
public:
     ...
     double Abstand(const Punkt& p);
};

class Gerade {
     ...
public:
     Gerade (Punkt p1, Punkt p2);      // g durch p1 und p2 (p1!=p2)
     Gerade (double pm, double pn);    // y=m*x+n
     friend Punkt Schnittpunkt (const Gerade& s1, const Gerade& s2);
};

class Strecke: Gerade {
     Punkt P1, P2;
public:
     Strecke (Punkt p1, Punkt p2);
     Gerade Mittelsenkrechte ();
};
```

Hinweis: 1. Man verwende zur Repräsentation von Geraden/Strecken die Parameter der Geradengleichung `y=m*x+n` und behandle den Sonderfall (zur *x*-Achse) senkrechter Strekken/Geraden separat! 2. Eine Senkrechte zur Geraden `y=m*x+n` hat den Anstieg `-1/m` !

2. *Binärbaum* **[20 Punkte]**

Definieren Sie eine (für die folgende Anwendung hinreichende) Nutzerschnittstelle `BinaryTree.h` eines abstrakten Datentyps »Binärbaum« zum Sortieren ganzer Zahlen und implementieren Sie diese in `BinaryTree.cxx`. Verwenden Sie dabei einen Knotentyp, der neben den Verkettungsinformationen im Baum jeweils ein Datum des Typs `int` aufnehmen kann. Für ein Objekt des Typs `BinaryTree` soll es mittels des Operators `+=` möglich sein, neue Elemente (sortiert) in den Baum aufzunehmen. Darüber hinaus soll es möglich sein, bei der Ausgabe eines Objektes des Typs `BinaryTree` eine aufsteigende Ausgabe aller im Baum enthaltenen Elemente zu erhalten.

```
class BinaryTree { ... }; ....

BinaryTree aTree;
aTree+= 5;
aTree+= 27;
aTree+= -3;

cout << aTree << endl; // liefert: -3 5 27
```

Entwerfen Sie ein Testprogramm `BTtest.cxx` für den Datentyp `BinaryTree` das von der Standardeingabe bis zu einer abschließenden Eingabe von `0` zyklisch Zahlen einliest, diese (`0` ausgenommen) in einen Baum einfügt und am Ende sortiert ausgibt. Geben Sie die notwendigen Übersetzungsschritte bis zum ausführbaren Programm (in einem von Ihnen gewählten Referenzsystem) an, um aus den drei Quelltexten das ausführbare Testprogramm zu erzeugen.

3. *Complex Calculator* **[20 Punkte]**

Implementieren Sie einen Datentyp »Komplexe Zahl« mit den arithmetischen Operationen Addition, Subtraktion, Multiplikation und Division. Entwerfen Sie dazu ein Schnittstellenmodul `Complex.h`, ein Modul `Complex.cxx`, sowie ein Anwendungsprogramm *complex calculator*, welches in der Lage ist, (korrekt gestellte) Aufgaben mit komplexen Zahlen zu lösen. Die Syntax der Aufgaben sei folgende.

```
aufgabe ::= '(' <complex> ')' <operator> '(' <complex> ')' .
complex ::= <real> [ '+i*' <imag> ] . (Imaginärteil optional)
operator ::= '+' | '-' | '*' | '/' .
real, imag ::= reelle Zahl entsprechend C++-Syntax .
```

Eine Benutzung des *complex calculator*‘s (cc) könnte wie folgt aussehen:

```
cc> (1+i*3)+(4+i*5)[enter]
 == 5+i*8
cc> (1+i*-3)*(1+i*-3)[enter]
 == -8+i*-6
cc> (5)/(2)[enter]
 == 2.5
cc> q[enter]                          soll das Programm beenden
```

Implementieren Sie dabei auch geeignete Ein- und Ausgabeoperationen für komplexe Zahlen. **Hinweis**: Mit Hilfe der `istream`-Memberfunktion `putback` kann man ein bereits gelesenes Zeichen in einen `istream` “zurückstellen“, d.h. es wird bei der nächsten Zeichenleseoperation erneut gelesen. Geben Sie die notwendigen Übersetzungsschritte bis zum ausführbaren Programm (in einem von Ihnen gewählten Referenzsystem) an, um aus den Quelltexten das ausführbare Programm zu erzeugen.

4. *Vektoren* [10 Punkte]

Implementieren Sie eine Klasse zur Repräsentation von `double`-Vektoren beliebiger Länge.

```
class Vektor {
     double *data;
     int n;
public:
     Vektor (int dim);  // legt einen Vektor der Dimension dim an
                        // alle Elemente sind mit 0 initialisiert
     double& operator[](int);         // Vektorelement am Index n
                                      // (sofern existent)
          ...
};
```

Folgende Operationen sollen außerdem für `Vektor`-Objekte definiert sein:

- Skalarprodukt von Vektoren gleicher Dimension;
- Zuweisung und Initialisierung von Vektoren, (ggf. mit Auffüllung von Nullen, wenn die Dimension des Quellvektors kleiner als die Dimension des Zielvektors ist; bzw. mit Ignorieren überzähliger Elemente im umgekehrten Fall)

Entwerfen Sie ein Schnittstellenmodul `Vektor.h` und die Implementation `Vektor.cxx`.

5. *Virtual Virtuality* [10 Punkte]

Welche Ausgaben erzeugt das folgende (korrekte) C++-Programm ?

```
#include <iostream.h>

#define VIRTUAL /*<----*/

#ifdef VIRTUAL
#define virtual virtual
#else
#define virtual
#endif

#define BODY(X,I)\
     int i;\
public:\
     X():i(I)  { cout<<' '<<#X<<"(): "; foo(); }\
     ~X()      { cout<<'~'<<#X<<"(): "; foo(); }\
     virtual void foo(){cout<<#X<<"::foo(), i=="<<i<<endl;}

class A { BODY(A,1) };
class B: public A { BODY(B,2) };
class C: public B { BODY(C,3) };

void bar(A& r){cout<<"bar():\t"; r.foo();}

main() {
     {    A a; bar(a);
          {    B b; bar(b); }
     }
     C c; bar(c);
}
```

Was ändert sich an den Programmausgaben, wenn die mit `/*<----*/` markierte Zeile gestrichen wird (das Programm bleibt dabei korrekt) ? **Hinweis**: Ein Makroparameter mit vorgestelltem `'#'` wird bei der Ersetzung zu einem String (*stringization*).

6. *Primzahlsummen* [10Punkte]

Die (immer noch unbewiesene !) *Goldbach*sche Vermutung aus dem Jahre 1742 besagt, daß sich jede gerade natürliche Zahl `n>=6` als Summe von zwei Primzahlen darstellen läßt. Schreiben Sie ein C++-Programm, welches für `n<=1000` diese Vermutung praktisch durch Angabe je eines Paares von Primzahlen mit `n=p1+p2` nachweist.

7. *Fehler über Fehler* [10Punkte]

Finden Sie möglichst viele Fehler in dem folgenden vermeintlichen C++-Programm, welches eine einfache »Summiermaschine« implementieren soll. Reelle Zahlen (Preise) in Fixpunktdarstellung sollen zeilenweise eingegeben und addiert werden, falls den Zahlen das Zeichen `'+'` folgt. Folgt das Zeichen `'='`, so soll die Summe aller bisher erfaßten Zahlen als Nettosumme und zusätzlich die Bruttosumme (incl. Mehrwertsteuer) ausgegeben und das Programm danach beendet werden. Eingaben, die mit einem anderen Zeichen abgeschlossen werden, sollen in der Summe ignoriert werden. Wird dem Programm beim Start ein Parameter übergeben, so soll neben den Ausgaben via `cout` ein Protokoll der Summation in eine Datei mit diesem Namen geschrieben werden. **Hinweis**: Die Klasse `ofstream` aus `fstream.h` besitzt einen Konstruktor `ofstream (const char* x)`, der den Ausgabestream an die Datei `x` knüpft.

```
/*  1 */ #include "iostream.h"
/*  2 */ #include <fstream.h>
/*  3 */
/*  4 */ /* eine Summiermaschine
/*  5 */ */
/*  6 */
/*  7 */ const MWSt = 15; //Prozent
/*  8 */ int logging;
/*  9 */ ofstream* log;
/* 10 */ double brutto (double netto; int Steuer=MWSt) (
/* 11 */      return netto * 1.0+MWSt/100;
/* 12 */ }
/* 13 */
/* 14 */ template <class TYP> // shorthand for out(int),out(double) and
/* 15 */                             out(const char*)
/* 16 */ void out(TYP d) {
/* 17 */      cout<<d;
/* 18 */      if logging log<<d;
/* 19 */ }
/* 20 */
/* 21 */ main(char**v, int c) {
/* 22 */    if (c>0) { log=new ofstream(v[1]); logging==1; }
/* 23 */
/* 24 */    double saldo; summand;
/* 25 */    char ch;
/* 26 */    while(1) {
/* 27 */      cin>>summand>>ch;
/* 28 */      switch (ch):
/* 29 */
/* 30 */      case = : saldo=+summand;
/* 31 */               out('=======\n'); out(saldo); out(" (Netto)\\n");
/* 32 */               out(brutto(saldo));
/* 33 */               out(" (incl. "); out(MwSt); out("% MwSt.)\n");
/* 34 */               return 0;
/* 35 */               break;
```

```
/* 36 */      case + :
/* 37 */                saldo+=summand;
/* 38 */      default:  out("(ignoriert!)\n");
/* 39 */                continue;
/* 40 */                break;
/* 41 */      }
/* 42 */     }
/* 43 */ }
```

8. » 2 hoch 100 « **[5Punkte]**

Schreiben Sie ein C++-Programm, welches den Wert von 2^{100} berechnet und ausgibt. Beachten Sie dabei, daß der Wert nicht innerhalb des Typs `long int` darstellbar ist.
Hinweis: `log 2 = 0.3010` !

9. Puzzle (*) **[5Punkte]**

Was gibt das folgende Programm aus?

```
#include <iostream.h>

class B {
public:
         B(const char* s) { cout << ' ' << s; }
};

#define D(s) class s:public B{public:s():B(#s){}}

D(It);         D(C);
D(Class);      D(With);
D(Do);

void x(){cout<<'!';}
void y(){cout<<"++"; x(); cout<<endl;}

class H : public Do {};
class I : public H, public It, public With {};
class P : public virtual I, public Class { public: P(){x(); } };
class Q : public I, public C {};
class R : public Q, public virtual P { public: ~R(){y();} };

int main() { R r; return 0; }
```

Hinweis:
Die Ausgabe dieses Programmes (incl. aller Leerzeichen) ist zugleich die *decryption phrase* für die Entschlüsselung der vorbereiteten Beispiellösungen unter

`ftp://ftp.informatik.hu-berlin.de/pub/local/vorlesung/c++/2.klausur.97/loesungen.asc`

Auspacken mittels:

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

tar xvzf loesungen
```