1.Klausur

zur Vorlesung

»Objektorientierte Programmierung in C++«

(Abschluss 1.Semester 1999/2000)

Bitte notieren Sie auf allen (!) Lösungsblättern Ihren Namen und ihre Matrikelnummer. Die Rückgabe der Ergebnisse und die Diskussion von Beispiellösungen erfolgt am 18.2.00. Mit der vollständigen Lösung der folgenden 7 Aufgaben können insgesamt 58 Punkte erreicht werden. Streben Sie bei der Auswahl der von Ihnen gelösten Aufgaben 20 Punkte an!

Viel Erfolg !!!

1.Kredit [14 Punkte]

Bankkredite aller Art müssen im Interesse der Vergleichbarkeit der Kreditkonditionen stets mit dem sogenannten effektiven Jahreszins ausgewiesen werden. Dabei handelt es sich sich um eine Verzinsung, die alle zusätzlichen Kreditkosten (incl. Nominalzins, Zinsezins und Gebühren) einschließt und die folgendermaßen erhoben wird:

Eine feste Rate ist über die gesamte Laufzeit des Kredites monatlich zu entrichten. In jedem Monat reduziert sich somit die Restschuld um eine Rate. Die jeweilige monatliche Restschuld wird zu 1/12 des effektiven Jahreszinses (== monatlicher Effektivzins) verzinst, wobei diese Zinsen über ein Jahr akkumuliert und jeweils am Jahresende (bzw. zu Ende der Kreditlaufzeit) auf die Restschuld aufgeschlagen werden. Die Rate wird so festgelegt, dass sie über die Kreditlaufzeit gerade den Bruttokreditrahmen (Nettokredite + Gesamtsumme aller anfallenden Zinsen) deckt, d.h. sei **m** die Laufzeit in Monaten, **N** die Nettokreditsumme, **B** die Bruttokreditsumme und **R** die monatliche Rate, so gilt:

B - R*m >= 0,00 DM und B - (R+0,01 DM)*m < 0,00 DM **[1]**

Da **B** selbst von **R** (nichtlinear) abhängt, gibt es keine geschlossene Formel zur Berechnung der Raten, also muss **R** approximativ so bestimmt werden, dass **[1]** gilt. Nimmt man z.B. *17000 DM* zu einem effektiven Jahreszins von *4,9%* über eine Laufzeit von *4 Jahren* in Anspruch, so ergibt sich nach approximierter Berechnung der Rate zu *389,83 DM* [1] der folgende Zahlungsplan:

```
lfd. Monat    Restschuld                Rate          kumulative Monatszinsen
 1             17000,00                 389,83            69,42 = (17000,00 * 4,9 / 1200)
 2             16610,17                 389,83           137,24 = 69,42 + (16610,17 * 4,9 / 1200)
 3             16220,34                 389,83           203,47 = 137,24 + (16220,34 * 4,9 / 1200)
.....
11             13101,70                 389,83           676,30 ....
12             12711,87                 389,83           727,94

neue Restschuld
               12711,87-389,83+727,94 = 13049,98

13             13049,98                 389,83            53,29 = (13049,98 * 4,9 / 1200)
.....
47               661,51                 389,83           117,26
48               271,68                 389,83           118,37       [271,68+118,37=390,05 !]


Nettokreditsumme:           17000,00
Bruttokreditsumme:          18711,84
Gesamtzinsen:                1711,84
Rate:                         389,83 (22 Pfennige Rest werden erlassen!)
```

Implementieren Sie eine Klasse **`Kredit`**, die bei Vorgabe der Kreditparameter *Nettokreditsumme*, *Kreditlaufzeit* und *effektiver Jahreszins* die Kreditbruttosumme und die monatliche Rate ermittelt.

1. bei einer Rate von 389,84 DM (ein Pfennig mehr) würden 31 Pfennig zuviel zurückgezahlt werden!

Hinweise:
1. Rechnen Sie mit Pfennigen (**`long`**) exakt, wo immer dies möglich ist.
2. Runden Sie, falls erforderlich kaufmännisch: [0, 0.5) abrunden [0.5, 1) aufrunden !
3. Vorschlag für die Struktur der Klasse **`Kredit:`**

```
class Kredit {
     long summe;
     double effZins;
     int monate;
     long brutto;
     long zinsen;
....
     long rate();

public:
     Kredit (long S, double Eff, int M)
          :summe(100*S), effZins(Eff), monate(M), zinsen(0) {}

     void calculate(); // gibt Rate und Bruttosumme aus
};

// Benutzung:
int main()
{
     Kredit k(17000, 4.9, 48);
     k.calculate();
}
```

2.1 aus 1000 [6 Punkte]

Schreiben Sie ein C++ -Programm, welches mit maximal 10 Fragen der Art:

`Ist die Zahl kleiner als X ? [j oder n]`

jede beliebige vom Benutzer ausgedachte Zahl zwischen 1 und 1000 bestimmt. Der Benutzer soll die Fragen wahrheitsgemäß mit '**`j`**' oder '**`n`**' beantworten. **X** steht dabei für jeweils vom Programm bestimmte Zahlen, mit denen der Suchraum geeignet eingeschränkt wird.

Hinweis: Verwenden Sie

```
class I {
     int lower, upper;
public:
     I(int l, int u):lower(l), upper(u){}

     void ask()
     {
     // stellt geeignete Fragen und benutzt ggf. neue (lokale) I-Objekte
     }
};

int main()
{
     I i(1, 1000); i.ask();
}
```

3.Key Concepts [10 Punkte]

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

```
#include <iostream>

#define O(X) std::cout<<X

struct A {
     A(int i=1)             { O(i);}
     ~A()                   { O(-1);}
     void foo()             { O(2); bar();}
     virtual void bar()     { O(-2);}
} a;

struct B: public A {
     B(int i=3): A(i+1)     { O(i);}
     ~B()                   { O(-3);}
     virtual void foo()     { O(4);}
     void bar()             { O(-4); foo();}
};

struct C: public B {
     C(int i=5): B(i+2)     { O(i);}
     ~C()                   { O(-5);}
     void foo()             { O(6); bar();}
     void bar()             { O(-6);}
};

int main()
{
     B& b = *new B;
     {
          C c;
          a.foo(); a.bar();
          b.foo(); b.bar();
          c.foo(); c.bar();
          A* p = &a;
          p->foo(); p->bar();
          p = &b;
          p->foo(); p->bar();
          p = &c;
          p->foo(); p->bar();
     }
}
```

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.00/loesungen.asc
```

Auspacken mittels:

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

4.Fehlertext [10 Punkte]

Finden Sie möglichst viele Fehler in dem folgenden vermeintlichen C++-Programm, welches die ersten 40 ***Fibonacci***-Zahlen berechnen und ausgeben soll. Dabei sollen alle bereits berechneten Zahlen in einem Feld als **`Fib`**-Objekte gemerkt und bei Bedarf wiederverwendet werden.

```
  #include "iostream"

  class Fib {

       long fn;

       static Fib** fibs;

       class Init;   /// there will be a local class Init

       struct Init {
       public:
            Init(int n)
            {
                 fibs = new Fib * [n];
                 for(j = 0, j < n, ++j)fibs[j]=0;
                 fibs[0] = fibs[1] = new Fib(1);
            }
       }

       static Init init;

  Fib(int n): {
       if (n==1)
            fn == 1;
       }

       if (fibs[n-1] = 0) fibs[n-1] = new Fib(n-1);

       fn = (*(fibs+n-1))-> fn + fibs[n-2]->fn;
  };

public:
  static int fib (int n)
  {
       typedef Fib * & F;
       F f = fibs[n];
       retrun f ? f->fn : (f = new Fib(n), f->fn);}
};

const int N = 40;

Fib* Fib::fibs;
Init Fib::init (N);

void main()
{
  for (int i = 0; i <= N; ++i)
  {
       cout >> fib (i) << endl;
  }
}
```

5. Das Orakel von ~~Delphi~~ C++ [6 Punkte]

Ein Orakel wird gefragt, welches das kürzeste C++ -Programm ist. Das Orakel lässt sich die Regeln von (Standard-)C++ erklären, und kann nun entscheiden, ob ein Programm richtig oder falsch ist. Um das kürzeste Programm zu ermitteln, probiert es der Reihe nach alle Zeichenfolgen aus (erst die kurzen, dann die längeren), und überprüft, ob sie ein gültiges C++-Programm darstellen. Es konstruiert dazu systematisch Zeichenfolgen in lexikographischer Ordnung aus den für C++ -Programme zulässigen Zeichen. Dies sind laut C++ -Standard die folgenden:

> **[lex.charset] 2.2 Character sets**
>
> 1 The *basic source character set* consists of 96 characters: the space character (0x20), the control characters representing horizontal tab (0x09), vertical tab (0x11), form feed (0x0c), and new-line (0x0a), plus the following 91 graphical characters:
>
> ```
> a b c d e f g h i j k l m n o p q r s t u v w x y z (0x62 ... 0x7a)
> A B C D E F G H I J K L M N O P Q R S T U V W X Y Z (0x41 ... 0x5a)
> 0 1 2 3 4 5 6 7 8 9                                 (0x30 ... 0x39)
> _ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " ’ (alle > 0x20)
> ```

Nach längerer Überlegung findet es eine Lösung.

a) Welche Antwort gibt das Orakel?

b) Angenommen, das Orakel kann jede Testzeichenfolge in einer Nanossekunde (10^{-9} s) erzeugen und prüfen. Wenn es jetzt sofort damit beginnt, wann ist die Lösung zu erwarten:

- innerhalb der nächsten Stunde	[A] ?
- in dieser Woche	[B] ?
- in diesem Jahr	[C] ?
- in diesem Jahrtausend	[D] ?
- später	[E] ?

6. Rot13 [6 Punkte]

Implementieren Sie ein C++ -Klasse, mit der es möglich ist, Zeichenketten zu kodieren und zu dekodieren. Verwenden Sie das einfache Verfahren ***Rot13***, welches manchmal bei *emails* oder *news* verwendet wird, um Texte *'auf den ersten Blick unlesbar'* zu machen. Das Verfahren basiert auf einer einfachen Verschiebung aller Buchstaben um 13 Positionen im Alphabet. Aus **a** wird dabei **n** und aus **S** wird **F**, alle Nicht-Buchstaben bleiben unverändert.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m

Hinweis: Sie können die folgenden Makros aus **<cctype>** verwenden:

```
int islower(int c); // ist c ein Kleinbuchstabe ?
int isupper(int c); // ist c ein Grossbuchstabe ?
```

Bauen Sie die Klasse so, dass folgendes Filterprogramm seine Eingaben ***Rot13***-kodiert ausgibt:

```
#include <iostream>
#include "Rot13.h"

int main()
{
     char ch;
     while (std::cin.get(ch)) std::cout<<Rot13::encode(ch);
}
```

Implementieren Sie außerdem ein Filterprogramm, welches eine ***Rot13***-kodierte Eingabe wieder zum ursprünglichen Klartext dekodiert.

7. String Caching [6 Punkte]

Geben Sie eine Klasse an, die Zahlen mittels der C-Funktion **sprintf** formatiert und in Zeichenketten umwandelt. **sprintf** hat im C++ -Headerfile <cstdio> folgenden Prototyp:

```
extern "C" int sprintf (char *s, const char* format, /* args */ ...);
```

Dabei werden die Argumente der variablen Parameterliste ... der Reihe nach entsprechend den Vorgaben in ***format*** interpretiert und ab ***s*** beginnend als (null-terminierte) Bytefolge abgelegt. Der Benutzer ist dafür verantwortlich, dass sich bei ***s*** eine hinreichend große Speicherfläche befindet. Der Rückgabewert von **sprintf** gibt die Anzahl der insgesamt abgelegten Bytes (ohne Nullbyte) an. **%d** als Teil des Formatstrings ***format*** wandelt das nächstfolgende Argument als Dezimalzahl in eine Zeichenkette um. Z.B. legt **sprintf (str, "%d", 123);** bei **str** die Zeichenfolge **"123\0"** ab.

Die Klasse sollte die folgende Schnittstelle besitzen.

```
class Int{
 public:
      Int(int value);
      char* toString();
};
```

Die Umwandlung in eine Zeichenkette sollte den Text nicht jedes mal neu erstellen, sondern sich das alte (d.h. das erstmalige) Ergebnis im betreffenden Objekt merken. Achten Sie auf eine ausgeglichene Speicherbilanz! Dies könnte insbesondere Probleme mit sich bringen, wenn es erlaubt ist, **Int**-Objekte zu kopieren. Lösen Sie diese Sie auf zwei verschiedene Weisen, indem Sie

a) verbieten, dass überhaupt Kopien von **Int**-Objekten erlaubt sind und
b) eine geeignete Kopiersemantik für **Int**-Objekte implementieren.