1. Klausur

zur Vorlesung

„Objektorientierte Programmierung in C++“

(Abschluß 1. Semester [elementares C++]) 1997/98

Bitte notieren Sie auf allen Lösungsblättern Ihren Namen und ihre Matrikelnummer. Die Bekanntgabe der Ergebnisse erfolgt anonymisiert ab 16.2.98 durch Aushang. Mit der vollständigen Lösung der folgenden 8(+1) Aufgaben können insgesamt 62(+4) Punkte erreicht werden. Streben Sie bei der Auswahl der von Ihnen gelösten Aufgaben 26 Punkte an!

Viel Erfolg !!!

---

1. Stack [10 Punkte]

Schreiben Sie zum abstrakten Datentype **`Stack`** *(of int)* mit folgender Schnittstelle

```
#include "StackImpl.h"

//---- Stack -----------------------------------------------------------
class Stack{
protected:
     StackImpl* rep;
     Stack(const Stack& other); // copy not allowed !
public:
     Stack(int dim=100): rep(new StackImpl(dim)) { }
     ~Stack()                { delete rep; }
     void push (int i)       { rep->push(i); }
     int pop()               { return rep->pop(); }
     int full()              { return rep->full(); }
     int empty()             { return rep->empty(); }
     void print()            { rep->print(); }
};
```

eine Implementationsklasse **`StackImpl`**, die auf einer dynamischen (einfach verketteten) Liste basiert.

---

2. Determinaten [10 Punkte]

Schreiben Sie ein C++-Programm, welches für 3x3-Matrizen die Determinate berechnet. Die Eingabe der Matrix erfolgt von der Standardeingabe in Form von 9 (korrekten) reellen Zahlen auf einer Zeile. Verwenden Sie zur Berechnung die sog. Entwicklung nach einer Reihe (bzw. entsprechend nach einer Spalte):

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \cdot \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \cdot \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \quad \text{und} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} = a \cdot d - b \cdot c$$

Der Wert einer Determinate ändert sich nicht, wenn zu einer Zeile (Spalte) ein beliebiges Vielfaches einer anderen Zeile (Spalte) addiert wird. Versuchen Sie, den Algorithmus so zu strukturieren, daß er sich leicht z.B. auf 4x4-Matrizen übertragen läßt.

3. Konstruktoren / Destruktoren [6 Punkte]

Welche Ausgaben erzeugt das folgende C++-Programm ?

```
#include <iostream.h>

class X {
public:
     X()                    {cout<<'1'; f();}
     X(const X&)            {cout<<'2';}
     ~X()                   {cout<<'3';}
     void f()               {cout<<'4'; f(1);}
     virtual void f(int)    {cout<<'5';}
};

class Y: public X {
public:
     Y()                    {cout<<'6'; f();}
     Y(const Y& y): X(y)    {cout<<'7';}
     ~Y()                   {cout<<'8';}
     virtual void f()       {cout<<'9'; f(2);}
     void f(int)            {cout<<'0';}
};

struct O {~O(){cout<<endl;} }o;

X x1;

void f(X& x) {cout<<'-'; x.f(0);}

int main()
{
     X  x2=x1;
     X& x3=x2;

     f(x3);

     Y  y1;
     Y  y2=y1;
     Y& y3=y2;
     f(y3);
}
```

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.98/loesungen.asc
```

Auspacken mittels:

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

4. Datum [8 Punkte]

Entwerfen und implementieren Sie eine Klasse zur Darstellung von Datumsangaben. Die Klasse sollte wenigstens über die folgende Schnittstelle verfügen:

```
class Date {
     ...
public:
     Date (int day, int month, int year); // Initialisierung
     enum locale {de,us};
     enum length {Short, Long};
     void print(locale format=de, length l=Short);
     // Short: de: 10.2.98 us: 2/10/98
     // Long: de: 10. Februar 1998 us: february, 2nd 1998

     void next(); // weiterschalten auf den nächsten Tag,
     // yy ist Schaltjahr, wenn durch 4 teilbar, kein Schaltjahr, wenn
     // durch 100 teilbar, aber wiederum Schaltjahr wenn durch 400 teilbar
};
```

Alle Daten und Operationen, die außerdem erforderlich sind, sollen ebenfalls lokal zur Klasse **`Date`** sein !

---

5. Fehlertext [10 Punkte]

Finden Sie möglichst viele Fehler in dem folgenden vermeintlichen C++-Programm, welches eine Primfaktorzerlegung einer per Programmparameter übergebenen **`long`**-Zahl ermitteln soll. Wo liegt das größte *performance leak* dieses Programmes ?

```
/*  1 */ #include <iostream.h>
/*  2 */ #include "stdlib.h"
/*  3 */
/*  4 */ #ifdef HAS_NO_BOOL
/*  5 */ typdef int bool;
/*  6 */ const true==1;
/*  7 */ const false==0;
/*  8 */ #endif
/*  9 */
/* 10 */ class Prime {
/* 11 */    long val; // the prime value
/* 12 */    Prime* next;// the next Prime
/* 13 */    Prime ();// calculates next Prime in next()
/* 14 */    Prime (long l): val(l), next(0) // only once (for 2) called
/* 15 */    static long last_known;  // the last prime calculated so far
/* 16 */    static bool isPrime (long n);
/* 17 */ public:
/* 18 */    static Prime *first;
/* 19 */    Prime* next();
/* 20 */    long val() {return val;}
/* 21 */ }
/* 22 */
/* 23 */ bool Prime:isPrime(long n) {
/* 24 */    Prime p=first;
/* 25 */    long f=p->val();
/* 26 */    for (;;) {
/* 27 */         if ((n%f)) retrun false; // no prime
/* 28 */         p=p->next();
/* 29 */         if (p && p->val() < n/2) continue;
/* 30 */         else return true;
/* 31 */    }
/* 32 */ }
```

```
/* 33 */ Prime* Prime::next() {
/* 34 */    // if there is no next prime
/* 35 */    // the next one is created
/* 36 */    return next ? next : (next = new Prime());
/* 37 */ }
/* 38 */
/* 39 */ Prime::Prime() {
/* 40 */    val=lastknown;
/* 41 */    do ++val while ( ! isPrime(val) );
/* 42 */ }
/* 43 */
/* 44 */ long Prime::last_known = 2;
/* 45 */ Prime* Prime::first=new Prime(Prime::last_known);
/* 46 */
/* 47 */ void factorial (long n) {
/* 48 */    Prime p=Prime::first;
/* 49 */    long f=p->val();
/* 50 */    while (f <= n)
/* 51 */    {
/* 52 */         if (!(n % f)) // is f a factor ?
/* 53 */         {
/* 54 */              cout<<'\t'<<f<<endl;
/* 55 */              factorial(n/f);
/* 56 */         }
/* 57 */         else // try the next prime
/* 58 */              f = (p=p->next()).val();
/* 59 */         }
/* 60 */    }
/* 61 */ }
/* 62 */
/* 63 */ int main(char[]* p; int n)
/* 64 */ {{{
/* 65 */         long n=atol(p[0]);
/* 66 */         cout<<n<<endl;
/* 67 */         Prime::factorial(n);
/* 68 */ }}}
```

---

6. Factory [6 Punkte]

Implementieren Sie ein Code-Muster (*pattern*), welches garantiert, daß von einer Klasse X mit einem Konstruktor **`X::X(int n);`** nur dynamische Objekte entstehen können (keine globalen und lokalen Instanzen, auch nicht als Kopien!) und das auch nur unter der Voraussetzung, daß n>0 gilt. Benutzen Sie dazu das sog. *factory*-Muster:

```
class X { ... int i; ... };
class X_factory {
     ...
public:
// build from scratch:
     ... create (int n) ... // n wird zum i des erzeugten X
// build from another X:
     ... create (const X&) ...
// destruction:
     ... destroy (...) ...
};
```

7. strstr [6 Punkte]

Reimplementieren Sie die C-Standardfunktion

```
const char* strstr (const char* s1, const char* s2);
```

in C++, die als Resultat die Position des ersten Auftretens von **s2** als Teilstring von **s1** zurückgibt, oder 0, falls **s2** kein Teilstring von **s1** ist. Ist **s2** der leere String (**""**), so wird **s1** zurückgegeben.

---

8. e [6 Punkte]

Für die mathematische Konstante **e** (Basis des natürlichen Logarithmus) sind die beiden folgenden Grenzwertsätze bekannt.

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n \qquad e = \lim_{n \to \infty} \sum_{i=0}^{n} \frac{1}{i!}$$

Welcher von beiden ist besser zur näherungsweisen Berechnung von **e** geeignet? Begründen Sie Ihre Entscheidung und implementieren Sie eine entsprechende Berechnungsfunktion für **e** in C++.

(*) 9.Zugabe [4 Punkte]

Warum gibt das folgende Programm zweimal **original** aus, obwohl bei der Übergabe des Parameters an **print** eine Kopie (-->**copy**) übergeben wird ? Welche Korrektur bringt die erwünschten Ausgaben (**original** und **copy**) ?

```
// Copyright © 1997, Gimpel Software
// http://www.gimpel.com
#include "iostream.h"

class X {
public:
     X( const X & ) { x = "copy\n"; }
     X( ) { x = "original\n"; }
     char *x;
};

class Y : public X
{ public:
     Y( const Y & arg )  { y = arg.y; }
     Y() { y = 0; }
     int y;
};

void print( Y arg ) { cout << arg.x ; }

void main()
{
      Y a;
      cout << a.x;
      print( a );
}
```