BERGISCHE UNIVERSITÄT
GESAMTHOCHSCHULE WUPPERTAL
GAUSS-STRASSE 20
42097 WUPPERTAL
(Korrespondenzanschrift)
42119 WUPPERTAL
(Lieferanschrift)
TELEX 8 592 262 bughw
TELEFAX (0202) 439-2901
TELEFON (0202) 439-1

Fachbereich 7

MATHEMATIK

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

e-mail: Juergen.Buhl@math.uni-wuppertal.de

Grundzüge der objektorientierten Programmierung

WS2001/2002 – Übungsblatt 13

Abgabetermin: 11. Februar 2002

Aufgabe 1. *Virtuelle Funktionen, 5 Punkte*

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

class CForm {
public:
   void ZeigeForm(void);
   virtual void Kopf(void) { cout << "Das ist eine Kopfzeile" << endl; }
   virtual void Koerper(void) { cout << " Das ist der Text" << endl; }
   virtual void Fuss(void) { cout << "Das ist die Fusszeile" << endl; }
};

void CForm::ZeigeForm(void) {
   Kopf();
   for (int Index = 0 ; Index < 3 ; Index++) Koerper();
   Fuss();
}

class CMeineForm : public CForm {
   void Kopf(void) { cout << "Das ist eine neue Kopfzeile" << endl; }
   void Fuss(void) { cout << "Das ist eine neue Fusszeile" << endl; }
};

int main() {
   CForm *ErsteForm = new CForm;
   ErsteForm->ZeigeForm();
   delete ErsteForm;

   ErsteForm = new CMeineForm;
   ErsteForm->ZeigeForm();

   return 0;
}
```

Bearbeiten Sie theoretisch, d.h. ohne das abgedruckte Programm einzutippen:

- Welchen Text gibt das oben abgedruckte Programm aus?
- Die Klasse `CForm` wird so abgeändert, dass aus der virtuellen Funktion `Kopf` eine rein virtuelle Funktion wird. Welche Zeilen des Hauptprogramms (Funktion `main`) müssen dadurch geändert werden? Geben Sie eine kurze Begründung an. Welchen Text gibt das Programm jetzt aus, wenn alle zu ändernden Zeilen des Hauptprogramms auskommentiert werden?

Aufgabe 2. *Vererbung mit Templates, 15 Punkte*
Im Rahmen eines Projektes bekommen Sie die Teilaufgabe zugeteilt, Template-Array-Klassen für Felder zu implementieren. Abzuliefern ist eine Headerdatei `array.h`, die in Anwendungsprogramme eingebunden werden kann. Nähere Details zu Ihrer Teilaufgabe können Sie der nachfolgenden Beschreibung entnehmen, in der die Anforderungen der späteren Benutzer gesammelt wurden (Pflichtenheft):

Eine Klasse `array` soll als Basis-Template für die Ableitung von vier weiteren Versionen dienen. Im Gegensatz zu normalen Feldern soll der Indexbereich nicht notwendigerweise bei 0 beginnen. Die tatsächlichen Indexgrenzen können mit den Elementfunktionen `lbound` und `ubound` nachgefragt werden, die Anzahl der Komponenten kann mit der Elementfunktion `len` ermittelt werden. Arrays können einander komplett zugewiesen werden. Beim Operator `=` kann einem Array auch eines mit anderen Grenzen zugewiesen werden, der alte Speicher wird dabei freigegeben. Die nichtöffentlichen Member `lower` und `upper` speichern den kleinsten bzw. größten Index. `num_elem` ist die Anzahl der Elemente, also `upper-lower+1`, wird aus Effizienzgründen aber mitgespeichert (Redundanz). `data` zeigt auf ein herkömmliches Feld, das die eigentlichen Daten speichert.
Ein Konstruktor `array(int n)` erzeugt ein Array mit `n` Komponenten, die von 0 bis `n-1` nummeriert sind. Dagegen können beim Konstruktor `array(int,int)` beliebige Unter- und Obergrenzen angegeben werden. Die Konstruktoren rufen eine private Funktion `allocate()` auf, die beim Versuch, ein Array mit nichtpositiver Komponentenanzahl anzulegen, eine Exception `error` mit dem Text `"illegal array size"` verursacht.
Mittels der Funktion `setlbound(int k)` läßt sich der Startindex auf den neuen Wert `k` setzen.
Der Index-Operator `[]` muß für den Zugriff natürlich noch eine Adreßumwandlung vornehmen (d.h. die Untergrenze subtrahieren). Zur Ausgabe soll ein entsprechender Ausgabeoperator `<<` zur Verfügung stehen.
Diese Basisklasse wird für die Ableitung der folgenden weiteren Klassen verwendet:

1. `checked_array<T>`, hier wird beim Zugriff auf die Komponenten zusätzlich der Index auf seine Gültigkeit überprüft. Im Fehlerfall wird eine Exception `"index out of bounds"` generiert.
2. `arith_array<T>` ist eine Version für einen arithmetischen Grundtyp `T`. Die Operatoren `+` und `-` müssen zur Verfügung stehen. Außerdem muß die Umwandlung von `(int)0` nach `T` möglich sein. Als Elementfunktionen stehen das Aufsummieren aller Elemente (`sum()`) und das Löschen, d.h. das Setzen aller Elemente auf 0 (`clear`) zur Verfügung. Es werden wie in `array` keine Indizes überprüft.
3. Das Template `checked_arith_array<T>` hat die arithmetischen Eigenschaften des Templates `arith_array`, überprüft aber zusätzlich Indizes wie `check_array<T>`. Bei der Ableitung von `check_array<T>` und

`arith_array<T>` soll die Klasse `checked_arith_array<T>` die Funktionalität der beiden Oberklassen erben. Datenmember der Klasse `array` sollen dabei **nicht** doppelt angelegt werden.

4. `sortable_array<T>` ist von `checked_arith_array<T>` abgeleitet und besitzt zusätzlich die Sortierfunktion `sort()`. Als Sortierkriterium wird das Vorzeichen der Differenz zweier Objekte verwendet.