Anmerkungen

- Sichern Sie ihre Berechnungen, so weit wie möglich gegen fehlerhafte Eingaben ab.
- Verwenden Sie sprechende Variablennamen und formatieren Sie Ihren Programmtext ordentlich.
- Erstellen Sie Inputprompts und Ausgaben, sodass das Programm für BenutzerInnen angenehm zu bedienen ist.
- Schreiben Sie Ihre Programm so, dass mehrere Berechnungen ausgeführt werden können, ohne das Programm immer wieder neu starten zu müssen.
- Testen Sie Ihre Programme und unterziehen Sie die Ergebnisse einer Prüfung (Schätzung des erwarteten Ergebnisses, bzw. im Notfall mit dem Taschenrechner prüfen).
- Mit * markierte Beispiele sind etwas anspruchsvoller

Aufgabe 1

Stellen Sie die Vector-Klasse aus der Übungseinheit zumindest so weit fertig, dass alle Methoden, die bei der Beschreibung der Funktionalität angegeben sind, korrekt arbeiten.

Aufgabe 2

Implementieren Sie alle in den Folien der Übungseinheit beschriebenen Zusätze für die Klasse Vektor

Aufgabe 3

Implementieren Sie beliebige Beispiele aus den Vorwochen, in denen string oder vector verwendet wurde, verwenden Sie Arrays anstelle der Klassen aus der Standard-Library (Sie müssen sinnvolle Maximalwerte für die Größen Ihrer Arrays wählen).

Aufgabe 4

Die Klasse N0 dient zur Darstellung von natürlichen Zahlen mit bis zu 70 Stellen (Interne Darstellung als char Array). Zusätzlich zum Wert enthält jedes Objekt der Klasse N0 auch einen Status, der festlegt ob der im char Array dargestellte Wert gültig ist, oder nicht. Implementieren und testen Sie zumindest die folgenden Methoden:

```
N0::N0() // mit 0 initialisieren
N0::N0(int)
N0 N0::operator+(N0)
N0 N0::operator-(N0)
N0 N0::operator*(N0)
N0 N0::operator/(N0)
void N0::print()
```

Falls bei den arithmetischen Operationen (+ - * /) Überläufe entstehen, so wird ein Objekt mit dem Status ungültig retourniert. Bei der Ausgabe wird für ungültige Objekte der String "ungültig" ausgegeben.

Aufgabe 5

Schreiben Sie eine Klasse Bigint, mit der ganze Zahlen beliebiger Größe bearbeitet werden können (interne Darstellung als Vektor von int-Werten mit jeweils maximal 6 Stellen). Die üblichen Rechenoperationen +, - und * sollen unterstütz werden, ebenso wie ganzzahlige Division / und Modulo %.

Aufgabe 6

Die Klasse SafeInt dient zur Darstellung von ganzen Zahlen. Die einzelnen Operationen sind dabei analog zum vordefinierten Datentyp int definiert. Im Unterschied zu int sollen SafeInt-Operationen weitgehend abgesichert werden. D.h. falls das Ergebnis einer SafeInt-Operation nicht definiert ist (Über-/Unterlauf, Division durch 0, etc.) soll das Ergebnisobjekt dies "wissen" und bei weiteren Operationen oder einer Ausgabe berücksichtigen. Implementieren und testen Sie zumindest die folgenden Methoden:

SafeInt::SafeInt() // mit 0 initialisieren
SafeInt::SafeInt(int)
SafeInt SafeInt::operator+(SafeInt)
SafeInt SafeInt::operator-(SafeInt)
SafeInt SafeInt::operator*(SafeInt)
SafeInt SafeInt::operator/(SafeInt)
void SafeInt::print()

Aufgabe 7

Die Klasse Vector dient zur Darstellung von Vektoren im R3. Implementieren und testen Sie zumindest die folgenden Methoden:

Vector::Vector() // mit (0,0,0) initialisieren
Vector::Vector(double, double, double)
Vector Vector::operator+(Vector)
Vector Vector::operator-(Vector)
double Vector::operator*(Vector) //Skalarprodukt
Vector Vector::operator/(Vector) //Vektorprodukt
double Vector::norm() //Betrag(Länge) void Vector::print()

Aufgabe 8

Implementieren Sie eine Klasse Vektor analog zu Aufgabe 7, allerdings für Vektoren beliebiger (im Konstruktor vorzugebender) Dimension. Die binären Operatoren können nur ausgeführt werden, wenn beide Operanden dieselbe Dimension haben.

Aufgabe 9

Implementieren Sie eine Klasse Kurs zur Verwaltung der TeilnehmerInnen einer Lehrveranstaltung. Realisieren Sie einen Konstruktor, mit dem der Titel der Lehrveranstaltung festgelegt werden kann und weiters Methoden zur Eingabe von Namen und Matrikelnummern für die einzelnen Studierenden, die über eine Ihnen zugewiesene Nummer (d.h. einen Index) identifiziert werden. Es muss (über geeignete Methoden) möglich sein, die Ergebnisse von (maximal 10) Beurteilungen (i.e. erreichte Punkte bei Tests) für einzelne Studierende zu speichern und den Durchschnitt der erreichten Punkte pro Person und pro Kurs zu errechnen. Definieren Sie eine geeignete Methode, die eine Ergebnisliste für den gesamten Kurs druckt.
Zum Testen der Klasse muss ein geeignetes Testprogramm erstellt werden.