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.
- Mit + markierte Besipiele sollen Aufgabenstellungen repräsentieren, wie sie in der Praxis an einen herangetragen werden. Die schwammige Definition der gewünschten Funktionalität ist Absicht. Treffen Sie geeignete, sinnvolle Annahmen, oder fragen Sie gegebenenfalls im Forum nach.

Aufgabe 1

Stellen Sie die Vector-Klasse aus der Übungseinheit mit allen definierten (Zusatz)Funtionalitäten fertig.

Aufgabe 2

Vervollständigen Sie die Implementierung des Bespiels mit Personen und Konten aus der Übung

Aufgabe 3 +

Eine Bank verwaltet die Daten Ihrer Kunden (Kundennummer, Adresse, Geburtsdatum, Telephonnummer), sowie deren Konten (Kontonummer, Kontoart, Kontostand). Mögliche Kontenarten sind Girokonto und Sparkonto. Schreiben Sie ein Programm, das die Möglichkeit bietet, Kunden neu zu erfassen, Konten anzulegen, bzw. zu löschen und Transaktionen wie Überweisungen, Ein- und Auszahlungen auf den verschiedenen Konten durchzuführen.

Aufgabe 4 +

Implementieren Sie eine Klasse Transaktion, die zur Speicherung von Transaktionen auf Bankkonten verwendet werden kann. Jede Transaktion soll ein Datum, eine Uhrzeit, die Bankleitzahlen und Kontonummern der beteiligten Konten, sowie den Betrag enthalten. Die Klasse Transaktion soll außerdem die Vergleichsoperatoren <, == und > anbieten, die es erlauben, das Datum der Transaktion mit einem bestimmten Stichtag zu vergleichen. Also, wenn tr eine Variable vom Typ Transaktion ist, und die entsprechende Transaktion am 5.3.2003 durchgeführt wurde, dann gilt tr=="5.3.2003" und

tr<"1.7.2003" und tr>"1.1.2003". Die Vergleichsoperatoren sollen außerdem auch verwendet werden, um den Betrag der Transaktion mit einem beliebigen Betrag zu vergleichen, bzw. um zwei Transaktionen (betragsmäßig) zu vergleichen.

Aufgabe 5 +

Für statistische Untersuchungen werden verschiedene Merkmale (Größe, Gewicht, Alter und IQ) von Personen gemessen. Entwerfen Sie die Klassen Person und Personenliste, die es erlauben, die Daten zu erfassen und die erfassten Personen nach einem beliebigen, vom Benutzer auszuwählenden Merkmal sortiert auszugeben. Implementieren Sie die beiden Klassen und ein zugehöriges Hauptprogramm, das zum Testen der Funktionalität geeignet ist.

Aufgabe 6 +

Ein einfacher Roboter besitzt drei Sensoren, die ihm melden, ob er mit der linken, der vorderen oder der rechten Seite eine Wand berührt. Der Roboter kann folgende Aktionen ausführen: Rotation um 90 Grad nach links, Rotation um 90 Grad nach rechts und Bewegung um eine Längeneinheit vorwärts. Der Roboter wird programmiert, indem in einer Liste mit 8 Einträgen, von denen jeder einem möglichen Gesamtzustand aller drei Sensoren entspricht, eingetragen wird, welche Aktionen in diesem Zustand durchzuführen sind. Dabei kann jeweils eine Rotation und/oder eine Vorwärtsbewegung erfolgen. Also zum Beispiel: links frei, vorne frei und rechts frei bewirkt eine Rotation nach links und einen Schritt nach vor. Schreiben Sie eine Klasse Labyrinth, die in einem zweidimensionalen Feld leere Felder und Wände enthalten kann und die es der Klasse Roboter erlaubt, abhängig von einer aktuellen Position und Ausrichtung die entsprechenden Sensorwerte zu bestimmen. Schreiben Sie eine Klasse Roboter, die es erlaubt, den Roboter zu programmieren und an einer beliebigen Stelle des Labyrinths zu starten. Können Sie den Roboter so programmieren, dass er seinen Weg durch ein beliebiges Labyrinth findet?

Aufgabe 7

Schreiben Sie eine Klasse String, die zur Repräsentation von Zeichenketten mit maximal 80 Zeichen verwendet werden kann.

Schreiben Sie

- die nötigen Konstruktoren, damit folgende Definitionen von Objekten möglich sind:

String a; // erzeugt ein leere Zeichenkette
String b="abc"; // erzeugt eine Zeichenkette mit "abc" als Inhalt

- Die Vergleichsoperatoren == und !=, die jeweils den Inhalt zweier Strings vergleichen.
- Den Operator +, der zwei Strings miteinander verknüpft (String("abc")+String("def")="abcdef"). BeiÜberschreiten der Maximallänge muss das Ergebnis entsprechend abgeschnitten werden.

- Den Operator -, der alle Aufteten eines Zeichens aus dem String entfernt (String("abcabc")-'a'="bcbc").

Schreiben Sie ein Hauptprogramm, das mindestens zwei Instanzen Ihrer Klasse verwendet und die Verwendung der verschiedenen Methoden illustriert.

Aufgabe 8

Schreiben Sie eine Klasse Menge, die Teilmengen der Menge {1,2,3, …,32} repräsentieren kann. Die Menge soll durch eine Integervariable beschrieben werden, wobei gilt, dass das entsprechende Bit genau dann gesetzt sein soll, wenn eine Zahl in der Menge enthalten ist. Hat die Integervariable also z.B. den Wert 10, so ist die dargestellte Teilmenge {2,4}. Schreiben Sie die Operatormethoden für +,- und *. Dabei soll + die mengentheoretische Vereinigung sein, - die Differenz und * der Schnitt. Alle Operatoren sollen auch funktionieren, wenn der rechte Operand ein Element (also eine Zahl zwischen 1 und 32) und keine Menge ist. In diesem Fall wird einfach die Operation ausgeführt, als würde es sich um eine entsprechende einelementige Menge handeln, z.B.:

```
 Menge m; //Leere Menge
m=m+3; // {3}
m=m+5; // {3,5}
m=m-3; // {5}
m*3; // {}
m*5; // {5}
```

Aufgabe 9

Implementieren Sie eine Klasse Menge analog zu Aufgabe 8. Verwenden Sie aber für die interne Repräsentation statt einer Integervariablen ein Feld von Integerwerten. Welche Art der internen Repräsentation ist 'besser'?

Aufgabe 10

Implementieren Sie eine Klasse Dictionary, die dazu dient, Objekte so abzuspeichern, dass sie über einen frei zu vergebenden Schlüsselwert wieder aufgefunden werden können. Alle eingetragenen Objekte müssen einer Klasse Object angehören. Folgendes Programm soll möglich sein:

```
 Object a,b;
Dictionary dict; //leeres dictionary
dict.add(a,"Objekt A");
dict.add(b,"Objekt B");
dict.search("Objekt C"); // nicht gefunden
dict.search("Objekt A"); // -> liefert das Objekt, das der Variablen 'a' entspricht, zurück
```

dict.remove("Objekt A"); // entfernt einen Eintrag
dict.search("Objekt A"); // nicht gefunden, da der Eintrag entfernt wurde

Für die interne Repräsentierung empfiehlt sich eine Hashtabelle (Google!).

Aufgabe 11

Quaternionen sind algebraische Objekte ähnlich den komplexen Zahlen. Allerdings bestehen Quaternionen aus vier Komponenten anstatt zwei. Sie werden in der Form a+bi+cj+dk notiert. Für die Multiplikation gelten folgenden Formeln:

i*i=j*j=k*k=-1
i*j=k
i*k=-j
j*i=-k
j*k=i
k*i=j
k*j=-i

Beachten Sie, dass die Multiplikation nicht kommutativ ist. Implementieren Sie eine Klasse Quaternion mit den Operatoren +,- und *, sowie mit einer 'print' Methode zur formatierten Ausgabe.

Aufgabe 12

Implementieren Sie eine Klasse Polygon, die zur Repräsentation von Polygonen im dreidimensionalen Raum verwendet werden kann. Verwenden Sie intern eine Liste von Eckpunkten des Polygons, wobei die Kanten jeweils von einem Punkt zu dessen Nachfolger in der Liste verlaufen. Der Anfangspunkt gilt in diesem Zusammenhang als Nachfolger des letzten Punktes, so dass eine geschlossene, geometrische Figur entsteht. Realisieren Sie folgende Methoden (die notwendigen Parameter und Typen der Retourwerte sind selbst zu überlegen):

- add //Hinzufügen eines zusätzlichen Eckpunktes
- zoom //Vergrößern oder Verkleinern der gesamten Figur um einen vorgegebenen Faktor
- translation //Verschieben der gesamten Figur um einen vorgegebenen Vektor
- ==, != //Vergleichsoperatoren. Zwei Polygone sind genau dann gleich, wenn sie die gleichen Punkte und Kanten enthalten. Der Startpunkt ist allerdings nicht von Belang.

Beispiel für Gleichheit bei Polygonen:
(1 0 0) - (0 1 0) - (0 0 1) == (0 1 0) - (0 0 1) - (1 0 0) != (0 1 0) - (1 0 0) - (0 0 1)

Aufgabe 13

Die Klasse Matrix dient zur Darstellung von 4*4 Matrizen mit double Werten. Implementieren und testen Sie zumindest die folgenden Methoden:

Matrix::Matrix() // mit 0-Matrix initialisieren
Matrix::Matrix(char) // mit Einheitsmatrix initialisieren (lauter Einsen in der Diagonale, sonst 0)
Matrix::Matrix(double, double, double, ..., double) //mit 16 double Werten initialisieren (die Punkte sind nur als Abkürzung zu verstehen und so syntaktisch nicht korrekt)
Matrix Matrix::operator+(Matrix)
Matrix Matrix::operator-(Matrix)
Matrix Matrix::operator*(Matrix) //Matrixprodukt
Matrix Matrix::operator*(double) //Produkt Matrix und Skalar
void Matrix::print()

Aufgabe 14

Wie Aufgabe 13, allerdings für Matrizen beliebiger Dimension. Was wäre eine gute Lösung für den Konstruktor, mit dem die Werte aller Matrixelemente initialisiert werden können?

Aufgabe 15

Implementieren Sie die Klasse Polynom, deren Objekte Polynome über $\mathbb{R}$ vom Grad $<=20$ repräsentieren. Es sollen die Operatoren +,-,* und / mit der üblichen Bedeutung verwendet werden können. Eine Methode 'print' zur formatierten Ausgabe des Polynoms wird natürlich ebenso benötigt.

Aufgabe 16 *

Wie Aufgabe 14, allerdings sollen die Elemente der Matrizen nicht reelle Zahlen sein, sondern komplexe.

Aufgabe 17 *

Iteratoren dienen dazu, Listen von Objekten in einer vorgegebenen Reihenfolge zu durchlaufen. Sie werden gerne dazu verwendet, um Algorithmen (wie z.B. Sortieralgorithmen) unabhängig von der internen Repräsentation der Objekte bzw. der Liste formulieren zu können. Schreiben Sie die drei Klassen Objekt, Liste und Iterator, die folgende Funktionalität bieten sollen: Objekt ist ein Stellvertreter für eine beliebige Klasse. Diese Klasse sollte für Testzwecke zumindest eine print Methode aufweisen, die eine eindeutig Identifizierung der Objekte ermöglicht. Liste soll Objekte in einem Array verwalten. Die Liste soll die Methoden add und insert aufweisen, die ein Objekt am Ende bzw. am Beginn der Liste einfügen können. Iterator soll einen Konstruktor haben, der den Iterator an den Beginn der Liste setzt, die Operatoren ++ und –, die zum Weiter- bzw. Zurückgehen in der Liste dienen und eine Methode obj, die das aktuelle Objekt retourniert. Unter der Annahme, dass die Liste mit den vier Objekten 'a';, 'b, 'c' und 'd' gefüllt ist, soll z.B. folgendes Programmstück die angeführten Ergebnisse liefern:

```
Iterator it;
it.obj().print(); //Ausgabe von 'a';
it++;
it.obj().print(); //Ausgabe von 'b';
it--;
it.obj().print(); //Ausgabe von 'a'
```

Aufgabe 18 *

Ein endlicher, regulärer Automat kann eine Menge verschiedener Zustände annehmen. Dabei gilt ein Zustand als Startzustand, in dem sich der Automat am Beginn befindet. Beliebig viele Zustände können zu Endzuständen erklärt werden. In jedem Zustand liest der Automat ein Zeichen ein und geht abhängig von diesem gelesenen Zeichen in den nächsten Zustand über. Welche Zeichen welchen Folgezustand bewirken, wird anhand einer Tabelle festgelegt. Führt das Zeichen, das gelesen wurde, nicht zu einem neuen Zustand (ist es also nicht in der Tabelle entsprechend angeführt), so wird das als ein Fehler betrachtet. Erreicht der Automat einen Endzustand, so wird die Zeichenkette aller Zeichen, die bis dahin eingegeben wurden als 'vom Automaten akzeptiert' betrachtet. Implementieren Sie eine Klasse Automat, die zur Simulation regulärer Automaten verwendet werden kann, z.B. (Diese Tabelle ist natürlich nur ein Beispiel, Sie muss für Ihre Klasse entsprechend 'programmierbar'; sein):

Zustand	Endzustand	Zeichen	Folgezustand
1(Start)	Nein	a	1
		b	2
2	Nein	b	2
		c	3
		d	4
3	Ja	a	1
4	Ja		

Folgende Buchstaben werden nach dem Start des Automaten eingegeben (die entsprechende Ausgabe ist jeweils in Klammer vermerkt, fehlerhaft eingegebene Zeichen werden einfach ignoriert.):

ad(Fehler)abbc(akzeptiert: aabbc)c(Fehler)abd(akzeptiert aabbcabd)

Beachten Sie, dass keine weiteren Zeichen mehr eingelesen werden, sobald der Zustand 4 erreicht wird, weil in der Tabelle keine Folgezustände für den Zustand 4 vorgegeben sind.