Hausübungen
zu Programmierung II (C++)
Wintersemester 2001/2002

Thomas Letschert

FH Giessen–Friedberg
25. September 2001

Vorbemerkung

Hausübungen können gruppenweise erarbeitet werden. Sie werden jedoch einzeln abgegeben, dabei muss dargelegt werden, dass jeder ihrer Aspekte beherrscht wird. Die Abgabe erfolgt durch persönliche Vorführung und Erläuterung zu Zeiten, in den Räumen und auf den Rechnern des Praktikums. Etwa eine Woche vor dem Klausurtermin wird bei Bedarf ein gesonderter Abgabetermin eingerichtet.

Klausurvoraussetzung: Die Lösung von mindestens *zwei* Hausübungen *bis spätestens eine Woche vor der Klausur* ist *Voraussetzung zur Klausurteilnahme*. Bitte melden Sie sich frühzeitig, wenn Sie die Klausurzulassung bereits auf andere Art erworben haben.
Bonuspunkte: Für jede Hausübung, *die bis zu ihrem angegebenen Termin* abgegeben wird, erhalten Sie *zwei* Bonuspunkte.
Späte Abgabe: Für jede Hausübung, *die bis spätestens zwei Wochen nach dem angegebenen Termin* jedoch mindestens eine Woche vor dem Klausurtermin abgegeben wird, erhalten Sie bei der Klausurbewertung *einen* Bonuspunkt.
Verrechnung der Bonuspunkte: Bonuspunkte werden bei der *nächsten* Klausur zu dieser Veranstaltung verrechnet.

Beginnen Sie frühzeitig! Es ist erlaubt das Skript, auch das zu Programmierung I, zu benutzen.

Hausübung 1 (Abgabe bis zum 16. November 2001)

Thematik: Klassendefinitionen, Kapselung, konkrete Datentypen, elementare Programmierfertigkeiten (mathem. Formeln in Programmcode umsetzen, Funktionen, Methoden etc.), elementare Arbeitsorganisation (Skript/Buch finden und lesen, mit Kommilitonen, Dozenten sprechen etc.)

Implementieren Sie die *konkreten Datentypen* `Punkt`, `Vektor` und `Gerade` sowie die Hilfsklasse `Vektor` zur Modellierung der Geometrie in der Ebene. Ihre Klassen sollen alle Eigenschaften konkreter Datentypen haben, alle Datenfelder aller Klassen müssen privat sein und die Klassen müssen so gestaltet werden, dass unter anderem folgendes Programmstück übersetzt und mit korrekten Ausgaben ausgeführt werden kann:

```
int main (){
  Gerade g1, g2, g3;
  cout << "Bitte die erste Gerade eingeben" << endl;
  cin >> g1;
  cout << "Bitte die zweite Gerade eingeben" << endl;
  cin >> g2;

  if (g1 == g2)
    cout << "g1 und g2 sind gleich\n";
```

```
    else if (g1 || g2)
      cout << "g1 und g2 sind parallel\n";
    else {
      Punkt sp = g1 * g2;
      cout << "g1 und g2 haben den Schnittpunkt ";
      cout << sp << endl;
    }

    if ( !(g1 || Gerade::x_Achse)){
      cout << "g1 schneidet die x-Achse im Punkt ";
      cout << (g1*Gerade (Vektor(0,0), Vektor (1,0))) << endl;
    } else
      cout << "g1 ist parallel zur x-Achse" << endl;

    if ( !(g1 || Gerade::y_Achse)) {
      cout << "g1 schneidet die y-Achse im Punkt ";
      cout << (g1*Gerade::y_Achse) << endl;
    } else
      cout << "g1 ist parallel zur y-Achse" << endl;

    if ( g1 || Gerade(Punkt(2.0,3.5),Punkt(4.0,5.0)) ) // Gerade durch 2 Punkte
      cout << "g1 ist parallel zur Gerade "
           << Gerade(Punkt(2,3),Punkt(4,5)) << endl;
  }
```

Hausübung 2 (Abgabe bis zum 21. Dezember 2001)

Thematik: Zeiger, Vererbung, Definition einer Klassenhierarchie, Textverarbeitung, Rekursion, Organisation eines komplexeren Programms

Definieren Sie eine Klassenhierarchie für Ausdrücke und schreiben Sie ein Programm zur deren Analyse. Ein Ausdruck ist definiert durch folgende Grammatik:

```
<Ausdruck>          ::= <PunktAusdruck>
                      | <VektorAusdruck>
                      | <GeradenAudruck>
                      | <OperatorAusdruck>
<VektorAusdruck>    ::= "("<float>","<float>")"
<PunktAusdruck>     ::= "P"<VektorAusdruck>
<GeradenAusdruck>   ::= "G("<PunktAusdruck> "+" "R"<VektorAusdruck>")"
<GeradenAusdruck>   ::= "G("<PunktAusdruck> "," <PunktAusdruck>")"
<OperatorAusdruck> ::= "["<Ausdruck> "*" <Ausdruck>"]"
                      | "["<Ausdruck> "||" <Ausdruck>"]"
                      | "["<Ausdruck> "==" <Ausdruck>"]"
```

Ihr Programm soll einen Ausdruck einlesen und seine Baumdarstellung erzeugen. Eine Baumdarstellung besteht aus verzeigerten Objekten vom richtigen Typ die die entsprechenden Informationen enthalten. Z.B:

```
  [G(P(1,1)+R(2,2)) || G(P(3,3)+R(4,4))]
```

führt zu einer Struktur wie in Abb.1. Bei fehlerhaften Eingaben bricht Ihr Programm mit einer Fehlermeldung ab. (Ihr Programm soll die Ausdrücke rein formal und ohne Rücksicht auf ihre Bedeutung analysieren.)

Hausübung 3 (Abgabe bis zum 18. Januar 2002)

Thematik: Definition einer fortgeschrittenen Klassenhierarchie mit Basis- und Umschlagklasse, QT, STL

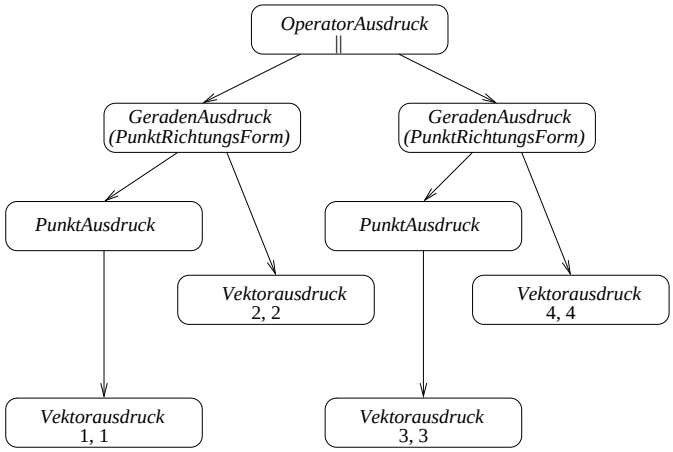

Abbildung 1: Zeigerstruktur eines Ausdrucks

Schreiben Sie auf Basis der Definitionen von Hausübung 2 und mit Hilfe von QT eine graphische Anwendung zur Auswertung von "geometrischen Zuweisungen". Bei einer geometrischen Zuweisung erhält eine Variable (hier ein Folge von Kleinbuchstaben) ein Objekt als Wert. Das Objekt wird durch den Ausdruck angegeben. Eine geometrische Zuweisung ist definiert durch:

```
<Zuweisung> ::= <Variable> "=" <Ausdruck>
<Variable>  ::= <Buchstabe>
              | <Buchstabe><Variable>
<Buchstabe> ::= "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"
<Ausdruck>  ::= ... wie oben ...
              | <VAusdruck>
<VAusdruck> ::= <Variable>
```

Ausdrücke können also auch Variablen enthalten. Die Interpretation der Operatoren ist:

- `||` Parallelität bei Geraden, undefiniert auf anderen Objekten
- `==` Gleicheit, definiert auf allen Objekten
- `*` Schnitt bei zwei Geraden;
  Test ob ein Punkt auf einer Geraden liegt, wenn der erste Operator eine Gerade und der zweite ein Punkt ist; ansonsten undefiniert.

Erweitern/modifizieren Sie zunächst Hausübung 2 derart, dass Zuweisungen mit dieser Syntax analysiert und in einen entsprechenden Baum umgesetzt werden können.

Definieren Sie dann auf Basis von Hausübung 1 eine Klassenhierarchie für geometrische Objekte (mit Basisklasse und Umschlagklasse) und schreiben Sie eine Funktion die einem Ausdruck einen Wert zuordnet. Achten Sie bei der Definition Ihrer Klassenhierarchie darauf, dass *jeder* syntaktisch korrekte Ausdruck einen Wert hat, auch wenn es ein boolescher Wert ist (z.B. Test auf Gleichheit etc.) oder er inhaltlich keinen Sinn macht (enthält Variable ohne Wert, Schnitt von 2 Punkten, etc.). Es muss also boolesche Werte und so etwas wie ein undefiniertes Objekt in der Klassenhierarchie geben.

Ihre Anwendung soll zwei Textfelder umfassen. In einem wird die Zuweisung eingegeben, in der anderen der Wert ihrer rechten Seite ausgegeben. Die Auswertung wird mit einem Knopf angestoßen, und mit einem anderen Knopf wird das gesamte Programm beendet. Bei der Auswertung wird der Wert der Variablen gespeichert.

Zur Speicherung der Variablenwerte verwenden Sie bitte eine von STL definierte Behälterklasse (Vektor, Liste oder Abbildung).

Hausübung 4 (Abgabe bis zum 18. Januar 2002)

Thematik: Dateien, Templatedefinition, Programmorganisation

1. Erweitern Sie Hausübung 3 um zwei weitere Knöpfe: *Laden* und *Speichern*. Mit *Speichern* werden die aktuellen Werte der Variablen in einer Datei gespeichert. Mit *Laden* werden sie aus der Datei geladen.
2. Ihr Programm muss entsprechend den üblichen Konventionen in Bestandteile zerlegt werden, die getrennt übersetzt werden können. Es muss durch eine Makedatei produziert werden.
3. Ersetzen Sie die STL–Behälterklasse durch ein selbst definiertes Klassentemplate.

Selbsteinschätzung

4 Aufgaben gelöst, keine verspätet abgegeben: Ausgezeichnet, fragen Sie nach einem Job als Tutor.

4 Aufgaben gelöst, maximal eine davon verspätet: Sehr gut.

3 Aufgaben gelöst, maximal eine davon verspätet: Gut.

3 Aufgaben gelöst: Nicht schlecht, damit kann man leben.

2 Aufgaben gelöst: OK, nicht jeder Informatiker muss Programmierer werden.

0–1 Aufgabe gelöst: Nicht OK, jeder Informatiker muss zumindest Chef von Programmierern werden können.