RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Lehrstuhl für Hochleistungsrechnen
RWTH Aachen · D-52074 Aachen · GERMANY
http://programmierung.informatik.rwth-aachen.de

Prof. Christian Bischof, Ph.D.
Andre Vehreschild, Jakob T. Valvoda

Übung *Programmierung WS 06/07* – Blatt 5

Lösungen müssen bis zum **27. November 2006, 17:00 Uhr** in den Kasten Ihrer Übungsgruppe eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Bitte vergessen Sie nicht, die **Nummer Ihrer Übungsgruppe**, sowie die **Namen** und **Matrikelnummern** der Mitglieder Ihrer 2er-Gruppe auf jedes Lösungsblatt Ihrer Abgabe zu schreiben. Bitte heften Sie ihre Blätter zusammen.

Alle erstellten Java-Programme sind sowohl in gedruckter Form abzugeben, als auch per E-Mail an den jeweiligen Tutor zu senden. Vergessen Sie auch bei der elektronischen Abgabe per E-Mail nicht die Angabe der **Nummer Ihrer Übungsgruppe**, sowie die jeweiligen **Namen** und **Matrikelnummern**.

Aufgabe 1 (4 Punkte)

In der Vorlesung wurden die Speicherarten Stack (Keller) und Heap (Halde) vorgestellt. Beantworten Sie die nachfolgenden Fragen:

a) Wie wird die Strategie genannt, nach der Datenblöcke (Frames) in den Stack eingefügt und entfernt werden?

b) In welchem Speicher werden Referenzen auf Arrays und Objekte gespeichert?

c) Welcher Speicher wird in Java vom „garbage collector“ verwaltet?

d) Kann die Anzahl der Elemente (also die Länge) eines Arrays nach der Speicherplatzzuweisung von n auf m (mit $n, m \in \mathbb{N}^+, m < n$) verkürzt werden? Falls ja, was geschieht mit den Elementen $m, \ldots, n-1$ welche nicht mehr erreicht werden können? Falls nein, mit welchen Mitteln könnte man eine Verkürzung nachbilden?

Diskutieren Sie die nachfolgenden Aussagen und geben Sie an, ob sie wahr oder falsch sind:

a) Mit `new` können Speicherblöcke auf dem Stack reserviert / alloziert werden.

b) Variablen für primitive Datentypen müssen vor ihrer Verwendung initialisiert werden.

c) Ein Array für einen beliebige Datentypen muss vor der Initialisierung immer deklariert werden.

d) Der „garbage collector“ muss immer eine Referenz auf das nicht mehr benötigte Objekt erhalten.

Aufgabe 2 (3 Punkte)

Implementieren Sie einen Algorithmus, welcher in einer Zeichenfolge nach dem ersten Vorkommen eines Teilwortes / Musters sucht. Sowohl die Zeichenfolge als auch das Muster sollen als Kommandozeilenparameter an das Programm übergeben werden. Verwenden Sie die Methoden `toCharArray()` um zuerst aus den `String`-Parametern Arrays vom Typ `char` zu generieren. Führen Sie auf diesen Arrays die Suche durch. Das Programm soll ausgeben, ob das Muster gefunden wurde und an welcher Position das Muster das erste Mal vorkommt.

Hinweis: Um einen String mit Leerzeichen als ein Parameter übergeben zu können, müssen Sie diesen in Anführungszeichen setzen. Beispiele:

```
>java Suche sting "a string searching example consisting of ..."
first occurence found at position 32

>java Suche sting "astrng stri st string"
input string does not contain 'sting'
```

Aufgabe 3 (4 + 4 + 2 + 2 = 12 Punkte)

In der letzten Übung haben Sie in Aufgabe 4 Methoden zum Zeichnen von Linien im Textmodus implementiert. In dieser Aufgabe soll der Algorithmus erweitert und zum Zeichnen beliebiger zweidimensionaler Polygone verwendet werden. Im Vordergrund steht hierbei die gedankliche Trennung zwischen geometrischen Daten und den Mechanismen zum Zeichnen auf dem Bildschirm.

a) Entwickeln Sie die Klasse `GraphicsContext` welche die Funktionalität der Zeichenfläche aus Aufgabe 4 a) der letzten Übung integriert.

   Verwenden Sie zur Darstellung der einzelnen Koordinaten der Zeichenfläche die Klasse `Vertex2D`, welche Sie von der Vorlesungs-Webseite herunterladen können. Ein Objekt vom Typ `Vertex2D` besitzt eine x- und eine y-Koordinate vom Typ `int`. Diese können Sie über die Methoden `getX()` und `getY()` auslesen und mit den Methoden `setX()` und `setY()` neu setzen.

   Die Klasse `GraphicsContext` verwaltet (analog zur letzten Übung) ein zweidimensionales Array `canvas` des Typs `char` als virtuelle Zeichenfläche. Definieren Sie zudem folgende Attribute und Initialisieren Sie diese mit Werten Ihrer Wahl:

   - `canvasWidth` und `canvasHeight` vom Typ `int`, um die Größe Ihrer Zeichenfläche festzulegen,
   - `brush`, `background`, `frame` vom Typ `char`, um jeweils ein Bildschirmzeichen für den Zeichenpinsel, den Hintergrund und den Rahmen der Zeichenfläche festzulegen.

Implementieren Sie folgende Methoden, um innerhalb des `GraphicsContext` malen und die Resultate anzeigen zu können (verwenden Sie für Methoden und Attribute weitgehend Quelltext aus Aufgabe 4 der letzen Übung):

- `void init( int, int )` – Zum Initialisieren soll die Höhe und die Breite des Bildschirms gesetzt werden. Die Methode legt ebenfalls die initialen Werte für `brush`, `background` und `frame` fest. Danach soll einmal `clearCanvas()` aufgerufen werden.
- `void setBrush( char )` – Zum Ändern des Vordergrundzeichens,
- `void setBackground( char )` – Zum Ändern des Hintergrundzeichens,
- `void setFrame( char )` – Zum Ändern des Rahmenzeichens,
- `void clearCanvas()` – löscht die virtuelle Zeichenfläche, indem das Array `canvas` komplett mit dem Wert des Attributs `background` und an den Rändern mit dem Wert des Attributs `frame` gesetzt wird,
- `void draw( Vertex2D )` – zeichnet die angegebene Koordinate in `canvas` mit dem Wert von `brush` ein. Hierbei soll überprüft werden, ob die Koordinate innerhalb der `canvas`-Zeichenfläche liegt. Modifikationen der Zeichenfläche sollten immer über diese Methode erfolgen.
- `void printCanvas()` – gibt den Inhalt von `canvas` auf dem Bildschirm aus.

b) In der letzen Übung haben Sie in Aufgabe 4 c) den Bresenham-Algorithmus kennengelernt und diesen für Geraden mit einer maximalen Steigung von 45° implementiert. Seien wiederum $p_s = (x_s, y_s)$ der Start- und $p_e = (x_e, y_e)$ der Endpunkt zwischen denen die Gerade gezeichnet werden soll. Sei ferner $\rho = \frac{y_e - y_s}{x_e - x_s}$ die Steigung der Geraden. Um nun beliebige Geraden zeichnen zu können, wird der Algorithmus in mehreren Schritten erweitert.

Geraden mit einer negativen Steigung bis $-45°$, d.h. $-1 \leq \rho < 0$, werden gezeichnet, indem in jedem Schritt zwischen den Punkten (x_i, y_{i-1}) und $(x_i, y_{i-1} - 1)$ entschieden wird. Die y-Koordinate wird also dekrementiert. Entsprechend wird die Fehlervariable angepasst, indem sie um den Wert 1.0 inkrementiert wird.

Alle anderen Geraden, also Geraden mit einer Steigung dem Betrag nach größer als 45°, d.h. $|\rho| > 1$, können gezeichnet werden, indem die x- und die y-Komponenten von p_s und p_e getauscht werden und mit diesen Punkten der obige Basis-Algorithmus ausgeführt wird. Beim Zeichnen müssen dann wiederum die Komponenten vertauscht werden.

Implementieren Sie den Bresenham-Algorithmus für beliebige Geraden als Methode `drawLine()` der Klasse `GraphicsContext`. Diese Methode erhält als Parameter zwei Objekte der Klasse `Vertex2D` und zeichnet die Gerade zwischen beiden Punkten in die Zeichenfläche. Für das Zeichnen einzelner Elemente soll `drawLine()` die Methode `draw()` aufrufen.

c) Polygone sind geometrische Objekte, welche aus mehreren aneinandergefügten Linien bestehen. Entsprechend werden Polygone durch eine Liste der Ecken, den sogenannten Vertices, repräsentiert.

Implementieren Sie die Klasse `Polygon` zur Darstellung von Polygonen. Diese Klasse soll ein `Vertex2D`-Array mit dem Namen `coordinates` als Attribut verwalten. Implementieren Sie die folgenden Methoden der Klasse

- `Vertex2D[] getCoordinates()` – Liefert eine Referenz auf das Array `coordinates` zurück,
- `void setCoordinates( Vertex2D[] coordinates )` – definiert das Polygon durch Übergabe einer Liste der Ecken.

Fügen Sie der Klasse `GraphicsContext` die Methode `drawPolygon()` hinzu, welche ein Objekt der Klasse `Polygon` als Parameter akzeptiert und es mit Hilfe der `drawLine()`-Methode ausgibt.

d) Schreiben Sie das Test-Programm `PolygonTestDrive`. Das Programm soll mindestens zwei Polygone mit jeweils mindestens drei Vertices erzeugen. Es ist Ihnen überlassen, ob Sie die Koordinaten vom Benutzer eingeben lassen, oder ob Sie diese im Quelltext fest verankern wollen.

Legen Sie dann ein Objekt des Typs `GraphicsContext` an, initialisieren Sie dieses, zeichnen sie das erste Polygon auf die virtuelle Zeichenfläche, wechseln Sie dann das Vordergrundzeichen mittels `setBrush()`, zeichnen Sie das zweite Polygon und geben Sie zum Schluss die Zeichenfläche mit `printCanvas()` aus.

Beispiel: Eine Ausgabe von `PolygonTestDrive`: